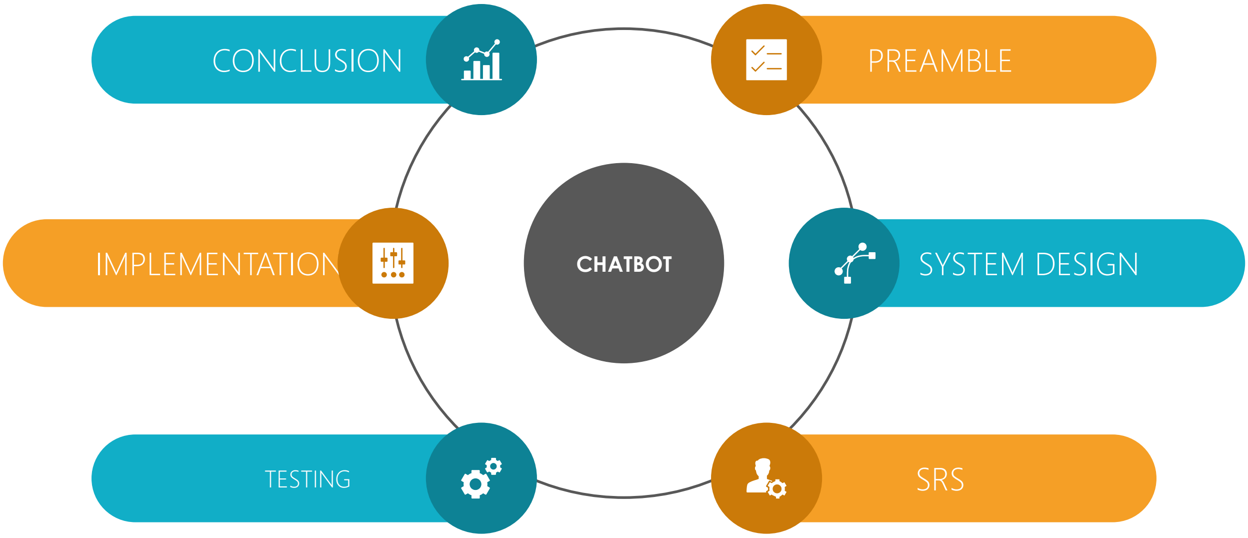
Compsoft technologies

AI HEALTHCARE CHATBOT

PRESENTED BY : Vignesh
Surya

PRESENTED TO: Compsoft
Technologies

Table of contents



CHATBOT

Abstract

Healthcare is very important to lead a good life.
However, it is very difficult to obtain the consultation with the doctor for every health problem. The idea is to create a medical chatbot using Artificial Intelligence that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. The chatbots are computer programs that use natural language to interact with users. The chatbot stores the data in the database to identify the sentence keywords and to make a query decision and answer the question

We will simply use pip to install the following :

- numpy
- nltk
- tensorflow
- tflearn

PROBLEM STATEMENT

Why a sudden need of this Chatbot?

Chatbots are services that people interact with through a messenger. Instead of having a conversation with another person, the user talks with a bot that's either powered by basic rules or machine learning. Every chatbot serves a specific purpose — health bots are designed to help with health-related Issues

Why is it best?

- Instantly 24/7 arability.
- A friendly humanly way interaction.
- Access anywhere, no matter where you are located.

This are the few advantages of this bot.

SRS

Functional Requirements:

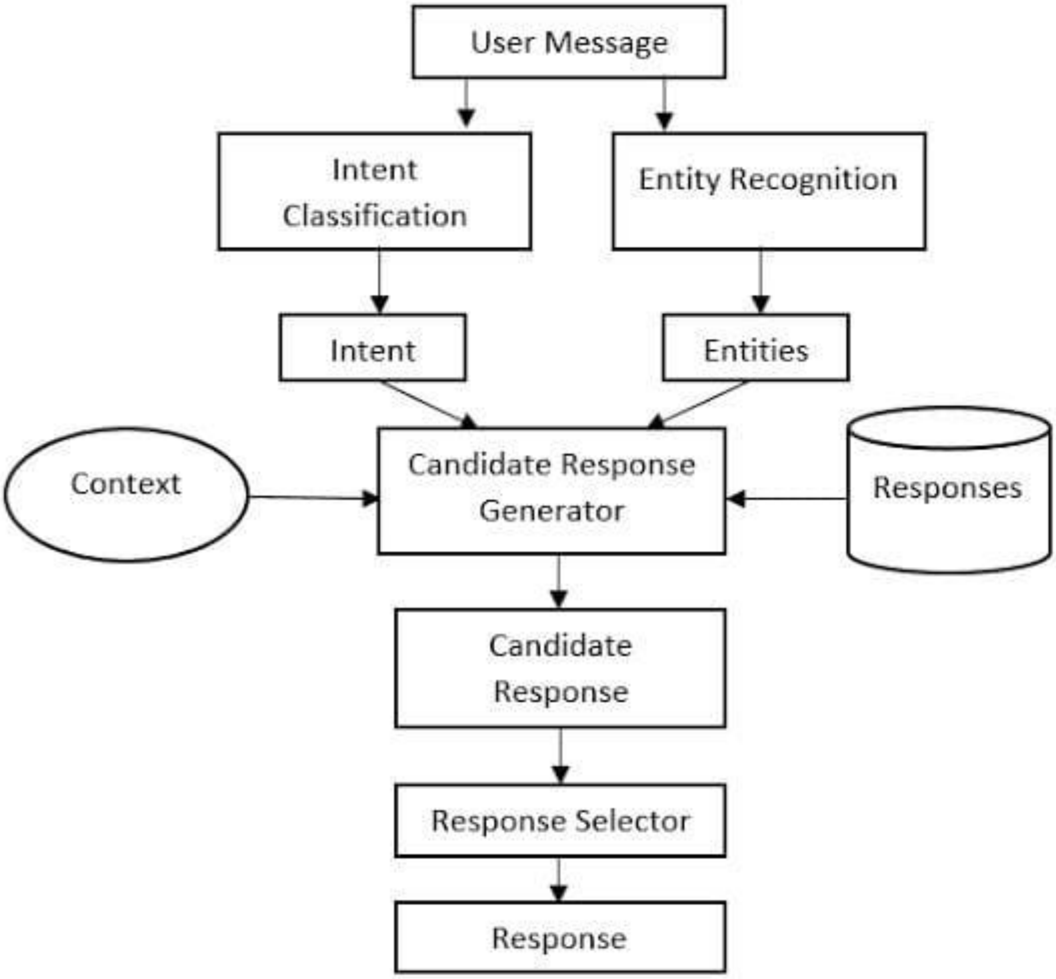
Hardware Requirements

- Pentium Processor IV or Higher
- Min 10 GB HDD
- RAM 512 MB or Higher
- 2.4 GHz or faster Processor

Software Requirements

- Windows Vista onwards, Linux, Mac OS
- In the case of building the Project from the source
- Python Compiler
- Tensorflow Machine learning library
- Keras
- SciKit Learn
- Pandas
- Numpy
- JSON

PROJECT ARCHITECTURE



Project Analysis

Loading our JSON DataWe will start by importing some modules and loading in our json data. Make sure that your .json file is in the same directory as your python script!

Extracting Data :Now its time to take out the data we want from our JSON file. We need all of the patterns and which class/tag they belong to. We also want a list of all of the unique words in our patterns

```
words = []
labels = []
docs_x = []
docs_y = []
```

loop through our JSON data and extract the data we want. For each pattern we will turn it into a list of words using nltk.word_tokenizer, rather than having them as strings. We will then add each pattern into our docs_x list and its associated tag into the docs_y list.

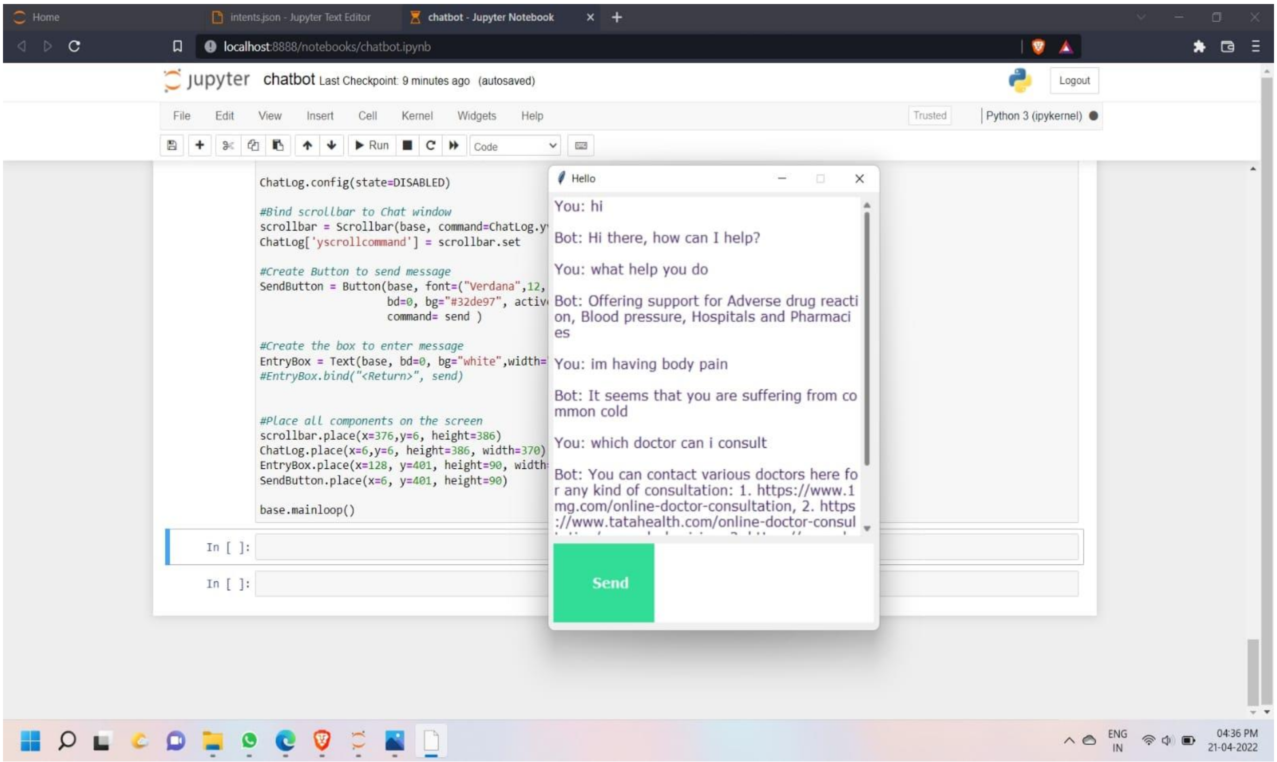
Making PredictionsNow its time to actually use the model! Ideally we want to generate a response to any sentence the user types in. To do this we need to remember that our model does not take string input, it takes a bag of words. We also need to realize that our model does not spit out sentences, it generates a list of probabilities for all of our classes. This makes the process to generate a response look like the following:– Get some input from the user– Convert it to a bag of words– Get a prediction from the model– Find the most probable class– Pick a response from that class

REFERRENCE

the references:

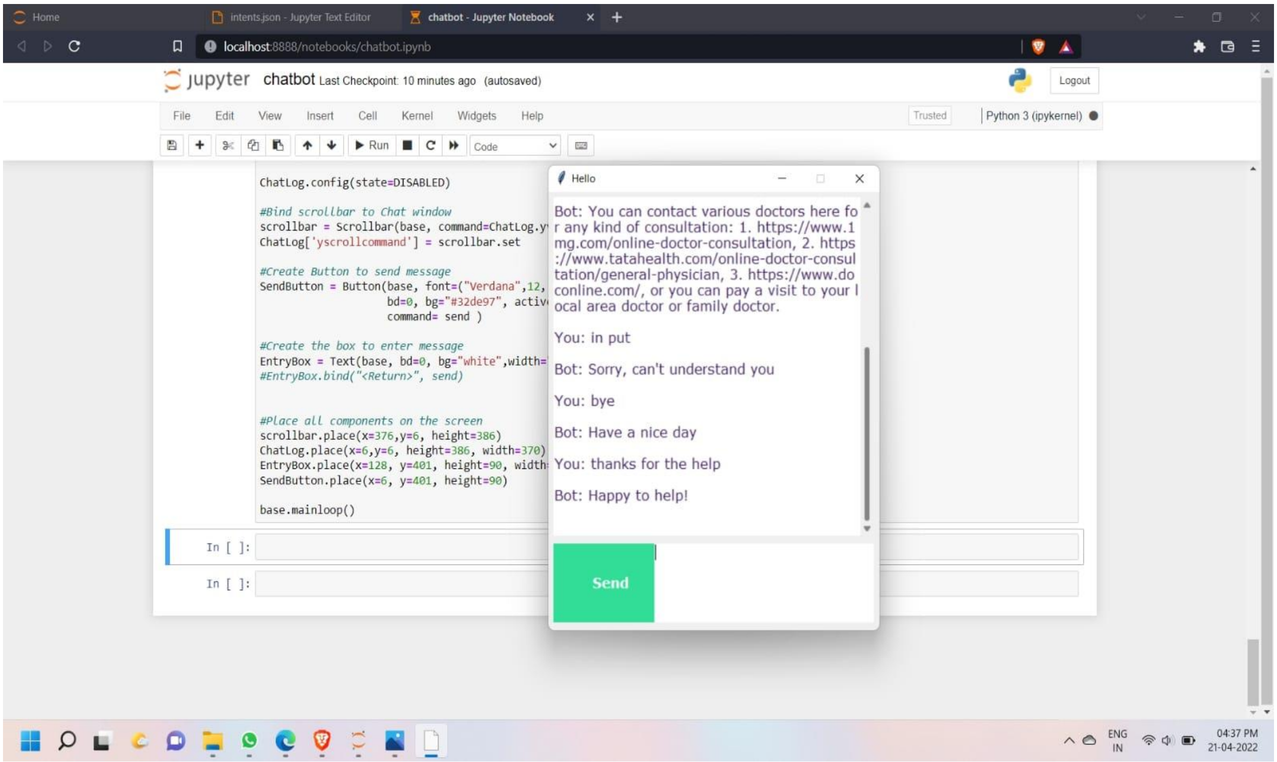
- https://www.wikipedia.org
- https://www.kaggle.com
- https://www.w3schools.com/
- https://github.com
- https://www.google.com

OUTPUT


Home
intents.json - Jupyter Text Editor
chatbot - Jupyter Notebook
localhost:8888/notebooks/chatbot.ipynb
jupyter
chatbot Last Checkpoint: 9 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
In [ ]:
In [ ]:
Hello
Send
ENG
IN
04:36 PM
21-04-2022

```
ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.y
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,
                    bd=0, bg="#32de97", activ
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width=
#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
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

```
You: hi

Bot: Hi there, how can I help?

You: what help you do

Bot: Offering support for Adverse drug reacti
on, Blood pressure, Hospitals and Pharmaci
es

You: im having body pain

Bot: It seems that you are suffering from co
mmon cold

You: which doctor can i consult

Bot: You can contact various doctors here fo
r any kind of consultation: 1. https://www.1
mg.com/online-doctor-consultation, 2. https
://www.tatahealth.com/online-doctor-consul
```


Home
intents.json - Jupyter Text Editor
chatbot - Jupyter Notebook
localhost:8888/notebooks/chatbot.ipynb
jupyter
chatbot Last Checkpoint: 10 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
In [ ]:
In [ ]:
Hello
Send
ENG
IN
04:37 PM
21-04-2022

```
ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.y
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,
                    bd=0, bg="#32de97", activ
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width=
#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
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

```
Bot: You can contact various doctors here fo
r any kind of consultation: 1. https://www.1
mg.com/online-doctor-consultation, 2. https
://www.tatahealth.com/online-doctor-consul
tation/general-physician, 3. https://www.do
conline.com/, or you can pay a visit to your l
ocal area doctor or family doctor.

You: in put

Bot: Sorry, can't understand you

You: bye

Bot: Have a nice day

You: thanks for the help

Bot: Happy to help!
```

INPUT

```
#import modules

from tkinter import *
import os

# Designing window for registration

def register():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("300x250")

    global username
    global password
    global username_entry
    global password_entry
    username = StringVar()
    password = StringVar()

    Label(register_screen, text="Please enter details below",
bg="blue").pack()
    Label(register_screen, text="").pack()
    username_lable = Label(register_screen, text="Username * ")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
    password_lable = Label(register_screen, text="Password * ")
    password_lable.pack()
    password_entry = Entry(register_screen, textvariable=password,
show='*')
    password_entry.pack()
    Label(register_screen, text="").pack()
    Button(register_screen, text="Register", width=10, height=1, bg="blue",
command = register_user).pack()
```

INPUT

```
# Designing window for login

def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("300x250")
    Label(login_screen, text="Please enter details below to login").pack()
    Label(login_screen, text="").pack()

    global username_verify
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
    global password_login_entry

    Label(login_screen, text="Username * ").pack()
    username_login_entry = Entry(login_screen,
textvariable=username_verify)
    username_login_entry.pack()
    Label(login_screen, text="").pack()
    Label(login_screen, text="Password * ").pack()
    password_login_entry = Entry(login_screen,
textvariable=password_verify, show= '*')
    password_login_entry.pack()
    Label(login_screen, text="").pack()
    Button(login_screen, text="Login", width=10, height=1, command =
login_verify).pack()
```

INPUT

```
# Implementing event on register button

def register_user():

    username_info = username.get()
    password_info = password.get()

    file = open(username_info, "w")
    file.write(username_info + "\n")
    file.write(password_info)
    file.close()

    username_entry.delete(0, END)
    password_entry.delete(0, END)

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

# Implementing event on login button

def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            login_sucess()
        else:
            password_not_recognised()
    else:
        user_not_found()
```

INPUT

```
# Designing popup for login success

def login_sucess():
    global login_success_screen
    login_success_screen = Toplevel(login_screen)
    login_success_screen.title("Success")
    login_success_screen.geometry("150x100")
    Label(login_success_screen, text="Login Success").pack()
    Button(login_success_screen, text="OK",
command=main_screen_success).pack()

# Designing popup for login invalid password

def password_not_recognised():
    global password_not_recog_screen
    password_not_recog_screen = Toplevel(login_screen)
    password_not_recog_screen.title("Success")
    password_not_recog_screen.geometry("150x100")
    Label(password_not_recog_screen, text="Invalid Password ").pack()
    Button(password_not_recog_screen, text="OK",
command=delete_password_not_recognised).pack()

# Designing popup for user not found

def user_not_found():
    global user_not_found_screen
    user_not_found_screen = Toplevel(login_screen)
    user_not_found_screen.title("Success")
    user_not_found_screen.geometry("150x100")
    Label(user_not_found_screen, text="User Not Found").pack()
    Button(user_not_found_screen, text="OK",
command=delete_user_not_found_screen).pack()
```

INPUT

```
#Deleting popups

def delete_login_success():
    login_success_screen.destroy()

def main_screen_success():
    main_screen.destroy()

def delete_password_not_recognised():
    password_not_recog_screen.destroy()


def delete_user_not_found_screen():
    user_not_found_screen.destroy()


# Designing Main(first) window

def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("300x250")
    main_screen.title("Account Login")
    Label(text="Select Your Choice", bg="blue", width="300", height="2",
font=("Calibri", 13)).pack()
    Label(text="").pack()
    Button(text="Login", height="2", width="30", command = login).pack()
    Label(text="").pack()
    Button(text="Register", height="2", width="30",
command=register).pack()

    main_screen.mainloop()


main_account_screen()
```

INPUT

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))


def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in
the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
```

INPUT

```
 # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result
def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res
```

INPUT

```
#Creating GUI with tkinter
import tkinter
from tkinter import *
def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        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, "Bot: " + res + '\n\n')

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

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#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)

base.mainloop()
```

Thank You