Health Assistant - Bot (Python)

A Major Project Report submitted to
Jawaharlal Nehru Technological University,
Hyderabad in partial fulfillment for the requirement
for the award of

Bachelor of
Technology
In
Computer Science & Engineering

BY

MD Shahid Hussain	**19645A0519**
S Indra kumar	**19645A0518**
K.Sai Kumar	**19645A0516**
T Prem Sai	**18641A05M9**
M.PAVAN	**19645A0517**

Under the Guidance of
B. Sravan Kumar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
VAAGDEVI COLLEGE OF ENGINEERING ,WARANGAL
(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")
Bollikunta, Khila Warangal (Mandal), Warangal Urban-506 005 (T.S)

JULY, 2022

VAAGDEVI COLLEGE OF ENGINEERING,WARANGAL

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban-506 005 (T.S)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project Report entitled Health Assistant - Bot (Python) is submitted by MD Shahid Hussain, S Indra kumar, K.Sai Kumar, T Prem Sai M. Pavan bearing RollNo(s) 19645A0519, 19645A0518, 19645A0516, 18641A05M9, 19645A0517 in partial fulfillment of the requirements for the award of the degree in Bachelor of Technology in Computer Science and Engineering during the academic year 2021-2022

GUIDE **Head of the Department**

External Examiner

ACKNOWLEDGEMENT

First and foremost, we sincerely thank our esteemed institution **Vaagdevi College of Engineering, Warangal** for giving us this great opportunity to fulfill our dreams of becoming Software Engineers.

We are highly obliged to the Head of the Department of Computer Science and Engineering, **Dr. N. Satyavathi** for her constant support and extensive help in each and every step.

We sincerely thank our Major project guide **B. Sravan Kumar**, Computer Science and Engineering for encouraging and supporting us throughout the project from start to end.

We would like to thank the Head of the Institution and Principal **Dr. K. Prakash** for the extensive support and for helping us in times we need it. Finally, we thank each and everyone who extended their help directly or indirectly and made this project successful.

MD Shahid Hussain (19645A0519)
S.Indra kumar (19645A0518)
K.Sai Kumar (19645A0516)
T.Prem Sai (18641A05M9)
M.PAVAN (19645A0517)

ABSTRACT

A chatbot is a computer software program that conducts a conversation via auditory or textual methods. This software is used to perform tasks such as quickly responding to users, informing them, helping to purchase products and providing better service to customers. Chatbots are programs that work on Artificial Intelligence (AI) & Machine Learning Platform. Chatbot has become more popular in business groups right now as it can reduce customer service costs and handles multiple users at a time. But yet to accomplish many tasks there is a need to make chatbots as efficient as possible. In this project, we provide the design of a chatbot, which provides a genuine and accurate answer for any query using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA) with python platform.

TABLE OF CONTENTS

Contents

CHAPTER-1 Introduction

1.1 Project purpose

The primary focus of this paper is to provide these services to fulfill the above mentioned purpose. It is difficult to imagine our lives without high tech gadgets because they have become an essential part of our lives.A chatbot is an automated software program that interacts with humans. A chatbot is merely a computer program that fundamentally simulates human conversations. A chatbot that functions through AI and machine learning has an artificial neural network inspired by the neural nodes of the human brain. Chatbots are programs that can do talk like human conversations very easily. For example, Facebook has a machine learning chatbot that creates a platform for companies to interact with their consumers through the Facebook Messenger application. In 2016, chatbots became too popular on Messenger. The software industry is mainly oriented on chatbots. Thousands of chatbots are invented on startups and used by the businesses to improve their customer service, keeping them hanging by a kind communication. According to research, nowadays chatbots are used to solve a number of business tasks across many industries like E-Commerce, Insurance, Banking, Healthcare, Finance, Legal, Telecom, Logistics, Retail, Auto, Leisure, Travel, Sports, Entertainment, Media and many others..

.

1.2 *Project* scope

This project can be developed even more by adding multilanguages, speech recognition. We can add many more tags to the data set, as the website gets developed. The chat history of a particular user can be sent as a mail to him/her after the conversation is over. This can be done by authorizing the users and receiving their mail id’s. This project is a small initiative to make the website user-friendly and easily understandable by the user.

The system is very useful for the companies or builders that can post and edit their properties and their personal info and admin can monitor records of all of them.The system is also useful which also keeps track of Account details of buyers and Investors and also RES Industry.

1.3 Project Objectives

- To extract symptoms from user chat
- To classify and predict the diseases using a decision tree classifier.

- To develop a healthcare chatbot to predict diseases by symptoms taken as input

1.4 Project Goals

- The goal of the project is to reduce man-power and to respond to user query at faster rate. Early days, the user's use to send a query mail to the particular site administrator and it would take few days for the site administrator to reply to the mails. Chatbots can overcome this delay, chatbot satisfies the user request or query immediately with relevant responses.

- A retrieval-based chatbot uses predefined input patterns and responses. It then uses some type of heuristic approach to select the appropriate response. It is widely used in the industry to make goal-oriented chatbots where we can customize the tone and flow of the chatbot to drive our customers with the best experience.

.

1.5 Existing system

- Healthcare Chatbot System Earlier, the artificial intelligence domain was not developed. After the invention of chatbot systems, the problems of users are solved in less time. In the field of healthcare, automated chatbot deployment in web applications is booming all over the world. Patients suffers from different types of diseases and visit to hospital for treatment purpose. Sometimes doctors are not available due to that, time required for nursing takes a lot [1].

  - To overcome this issue, medical chatbots were developed. These chatbots are trained and tested on live dataset also accuracy of the output is relevant. The AI based chatbot are fast, reliable and precise.

  - User provide the proper details and receive feedback according to their query. If any user makes minor mistake, the chatbot provides validation and autocorrection features.

  - Nowadays, in every clinics and hospitals portal chatbots are performing multitasking work. A lot of time of patient is saved and tasks are completed in minimum effort.

- Natural Language Processing (NLP) is a domain of artificial intelligence that provides machines to learn, read and understand the meaning of human languages. Different fields such as Banking, Education, Finance and so on are using chat applications for solving their problems and marketing their products.

- Majority of the countries like Germany, Spain, China, India, Korea and so on have their own native language. While interacting with other user, language barrier is created.

- To reduce this type of complication NLP based autonomous provides variety of languages, which makes communication through verbal and nonverbal easier [2].In the healthcare chatbot, NLP is used for text processing

- The characteristics of NLP in medical domain is useful:

    1. SentenceTokenization: In this method whole sentence is divided is into substrings and breaks into smaller words. When a patient enters the query in form of sentences, a whole set of words are converted into tokens, also split the sentences when there is punctuation mark.
    2. Word Tokenization: There are varieties of words, whichare present in the dictionaries.The segmentation of sentences is carried out. Words are assigned to tokens. The chatbot helps to classify the words according to the category present in medical records and gives the optimum feedback to a user.
    3. Stemming and Lemmatization: Stemming is process,where words are chopped out from beginning and end. Whereas, lemmatization is used in morphological analysis for extraction of words. When doctors entered the wrong data on their database, the responses of chatbot are not relevant. These above techniques are used to correct the information and give proper response to a patient in form of text.

1.6 PROPOSED SYSTEM

- The proposed method for developing the systemconsist of web application. Firstly, chatbot is createdwhich can help the users to get the symptoms of theirdiseases.

- Then we will add the chatbot link over therespective hospital website which will help the otherpeople to gain the information of medical reports

- Database of the system helps to store therecords of the users.

- We had train a chatbot using chatterbotlibrary and also train the bot to identify certain typesof keywords in order to recognize the user's intent.This information shall then be forwarded to thebackend

- The backend is responsible to use the processed input from the chatbot and convert it into action to be performed in the database.

- The healthcare chatbot is designed by using python in backend and user interface design by HTML, CSS and JavaScript.

- For conversation between user and system the natural processing library is used named chatterbot

- The bot also able communicate using a SpeechSynthesis.speak() Browser API

- The application runs in localhost server which provides appropriate details according to the user queries.

- All the database files are in yml format which are trained in the initial stage of the application model

1.6 Software Requirements

- The software Requirements in this project include:
- Python
- Anaconda prompt
- Spyder IDE
- Modules :
    1. Tensorflow

2. Keras
3. Pickle
4. Nltk
5. Flask

INSTALL SPYDER AND ANACONDA THEN

```
conda create -n spyder -env -y
conda activate spyder-env
conda install spyder-kernels scikit-learn -y
```

```
pip install tensorflow
pip install keras
pip install pickle
pip install nltk
pip install flask
```

```
run traning.py
run app.py
```

after running app.py in spyder copy the link

http://127.0.0.1:5000/

and paste the link in the browser.

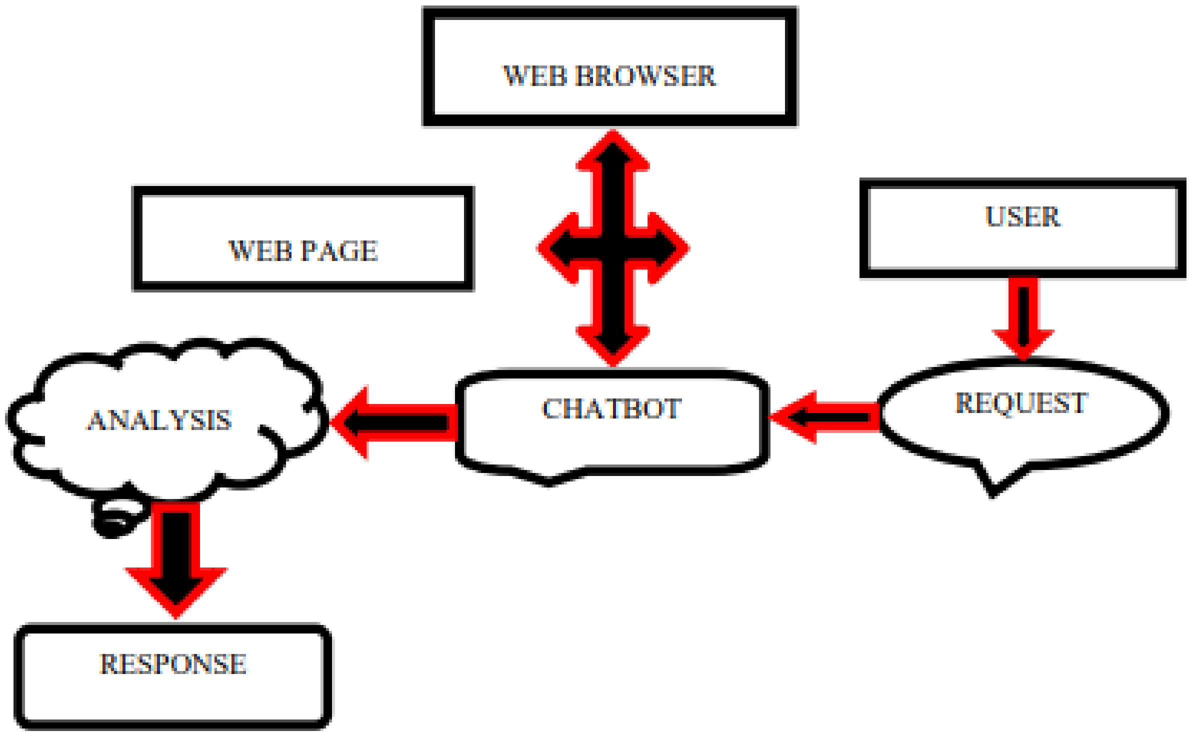
1.6 Hardware Requirements

The software Requirements in this project

1. Intel Processor
2. Webcam
3. Harddisk Above 256Gb
4. RAM Above 4Gb

CHAPTER-2 Design of the Project

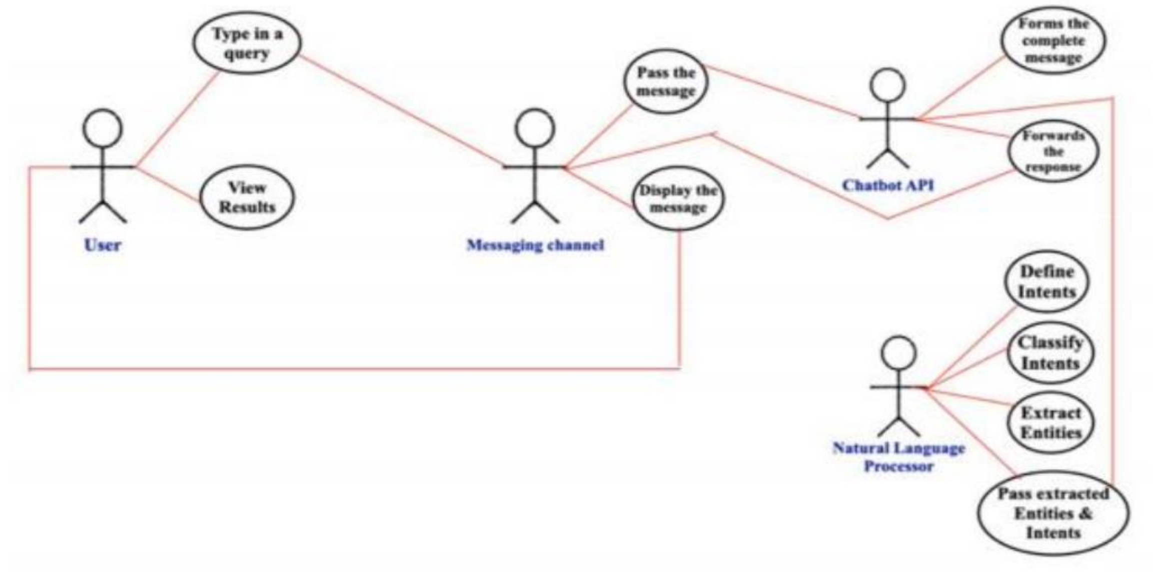
2.1 Flow Chart: The main purpose of this chatbot is to respond to user queries without man power. User can work with the chatbot in any web browser.



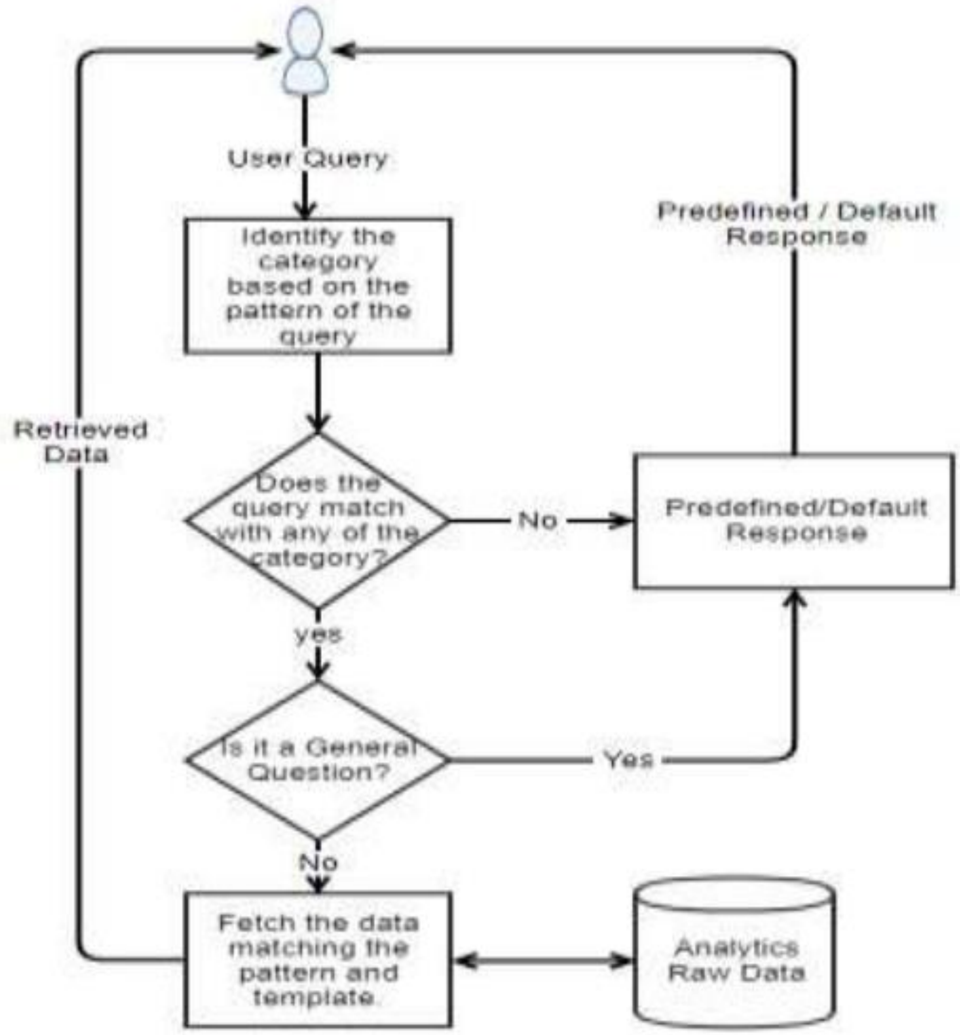
The chatbot receives the request sent by the user, analyses it and responds to the user in return. This analysis is done, using the machine learning algorithm.

The queries are predefined with a particular tag for each set. This tag is nothing but a keyword that helps the chatbot to analyze the user request. After the analysis, the chatbot responds to the user with required reply. If the user request is not clear to the chatbot, the response will be a default message defined by the developer. Almost all the queries from the user will be clearly responded, only few are exceptional cases.

2.2 Diagram



A chatbot is a computer program, which is designed to simulate a conversation with human users using patterns, especially over the internet. They are our online assistants that offer different services through chatting over the internet. To build artificial intelligence chatbots through Python, you will require an AIML package (Artificial Intelligence Markup Language). First, we need to create a standard startup file without any pattern and load aiml b in the kernel. Add random response patterns that would make dialogue interesting.


User Query
Identify the category based on the pattern of the query
Does the query match with any of the category?
No
yes
Predefined/Default Response
Predefined / Default Response
Is it a General Question?
Yes
No
Fetch the data matching the pattern and template.
Analytics Raw Data
Retrieved Data

CHAPTER-3 Implementations

This section describes the working of the system on an overall basis and further with specific focus on the software part of the chatterbot and the predefined query data set. An algorithm of the process, proceeded by a design motive of the system is also included.

3.1 File Structure

Data.json : The data file which has predefined patterns and responses.

Traning.py : In this Python file, we wrote a script to build the model and train our chatbot.

Texts.pkl : This is a pickle file in which we store the words Python object that contains a list of our vocabulary

Labels.pkl : The classes pickle file contains the list of categories.

Model.h5 : This is the trained model that contains information about the model and has weights of the neurons

We also create a list of classes for our tags. Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

3.2 Setup Flow

1. Import and load the data file

2. Preprocess data

3. Create training and testing data

4. Build the model

5. Predict the response

6. 1. Import and load the data file

7. First, make a file name as train_chatbot.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project.

8. The data file is in JSON format so we used the json package to parse the JSON file into Python

Preprocess data:

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Based on the requirements we need to apply various operations to preprocess the data. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words. Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list.

3.3 Traning.py

```
import nltk

nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import json

import pickle


import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from tensorflow.keras.optimizers import SGD

import random


words=[]

classes = []

documents = []

ignore_words = ['?', '!']

data_file = open('data.json').read()

intents = json.loads(data_file)



for intent in intents['intents']:

    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)


pickle.dump(words,open('texts.pkl','wb'))
pickle.dump(classes,open('labels.pkl','wb'))


# create our training data

training = []

# create an empty array for our output

output_empty = [0] * len(classes)

# training set, bag of words for each sentence

for doc in documents:

    # initialize our bag of words

    bag = []

    # list of tokenized words for the pattern

    pattern_words = doc[0]

    # lemmatize each word - create base word, in attempt to represent related words

    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]

    # create our bag of words array with 1, if word match found in current pattern

    for w in words:

        bag.append(1) if w in pattern_words else bag.append(0)


    # output is a '0' for each tag and '1' for current tag (for each pattern)

    output_row = list(output_empty)

    output_row[classes.index(doc[1])] = 1


    training.append([bag, output_row])

# shuffle our features and turn into np.array

random.shuffle(training)
```

```
training = np.array(training)

# create train and test lists. X - patterns, Y - intents

train_x = list(training[:,0])

train_y = list(training[:,1])

print("Training data created")



# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains
number of neurons

# equal to number of intents to predict output intent with softmax

model = Sequential()

model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train_y[0]), activation='softmax'))


# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this
model

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model

hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)

model.save('model.h5', hist)


print("model created")
```

3.4 App.py

```
import nltk

nltk.download('popular')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np


from keras.models import load_model

model = load_model('model.h5')

import json

import random

intents = json.loads(open('data.json').read())

words = pickle.load(open('texts.pkl','rb'))

classes = pickle.load(open('labels.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:
                # 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



from flask import Flask, render_template, request


app = Flask(__name__)
app.static_folder = 'static'


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


@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)




if __name__ == "__main__":
    app.run()
```

3.5 Data.json

```
{"intents": [
  {"tag": "greeting",
   "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
   "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
   "context": [""]
  },
  {"tag": "goodbye",
   "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
   "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
   "context": [""]
  },
  {"tag": "thanks",
   "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
   "responses": ["Happy to help!", "Any time!", "My pleasure"],
   "context": [""]
  },
  {"tag": "noanswer",
   "patterns": [],
   "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
   "context": [""]
  },
  {"tag": "options",
   "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you
can be helpful?", "What support is offered"],
   "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Hospitals
and Pharmacies", "Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies"],
   "context": [""]
  },
  {"tag": "adverse_drug",
   "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a list
of drugs causing adverse behavior", "List all drugs suitable for patient with adverse reaction", "Which
drugs dont have adverse reaction?" ],
   "responses": ["Navigating to Adverse drug reaction module"],
   "context": [""]
  },
  {"tag": "blood_pressure",
   "patterns": ["Open blood pressure module", "Task related to blood pressure", "Blood pressure data
entry", "I want to log blood pressure results", "Blood pressure data management" ],
   "responses": ["Navigating to Blood Pressure module"],
   "context": [""]
  },
  {"tag": "blood_pressure_search",
   "patterns": ["I want to search for blood pressure result history", "Blood pressure for patient", "Load
patient blood pressure result", "Show blood pressure results for patient", "Find blood pressure results by
ID" ],
   "responses": ["Please provide Patient ID", "Patient ID?"],
   "context": ["search_blood_pressure_by_patient_id"]
  },
  {"tag": "search_blood_pressure_by_patient_id",
   "patterns": [],
   "responses": ["Loading Blood pressure result for Patient"],
   "context": [""]
  },
  {"tag": "pharmacy_search",
   "patterns": ["Find me a pharmacy", "Find pharmacy", "List of pharmacies nearby", "Locate
pharmacy", "Search pharmacy" ],
   "responses": ["Please provide pharmacy name"],
   "context": ["search_pharmacy_by_name"]
  },
  {"tag": "search_pharmacy_by_name",
   "patterns": [],
   "responses": ["Loading pharmacy details"],
   "context": [""]
  },
  {"tag": "hospital_search",
   "patterns": ["Lookup for hospital", "Searching for hospital to transfer patient", "I want to search
hospital data", "Hospital lookup for patient", "Looking up hospital details" ],
   "responses": ["Please provide hospital name or location"],
   "context": ["search_hospital_by_params"]
  },
  {"tag": "search_hospital_by_params",
   "patterns": [],
   "responses": ["Please provide hospital type"],
   "context": ["search_hospital_by_type"]
  },
  {"tag": "search_hospital_by_type",
   "patterns": [],
   "responses": ["Loading hospital details"],
   "context": [""]
  }
]
}
```

3.6 index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Chatbot</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="{{ url_for('static', filename='styles/style.css') }}">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>

<body>
  <!-- partial:index.partial.html -->
  <section class="msger">
    <header class="msger-header">
      <div class="msger-header-title">
        <i class="fas fa-plus-square"></i> Health Care Bot <i class="fas fa-plus-square"></i>
      </div>
    </header>

    <main class="msger-chat">
      <div class="msg left-msg">
        <div                    class="msg-img"                    style="background-image:
url(https://image.flaticon.com/icons/svg/327/327779.svg)"></div>

        <div class="msg-bubble">
          <div class="msg-info">
            <div class="msg-info-name">HealthBot</div>
            <div class="msg-info-time">12:45</div>
          </div>

          <div class="msg-text">
            Hi, welcome to ChatBot! Go ahead and send me a message. 😄
          </div>
        </div>
      </div>

    </main>

    <form class="msger-inputarea">
      <input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">
      <button type="submit" class="msger-send-btn">Send</button>
    </form>
  </section>
  <!-- partial -->
  <script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
  <script>

    const msgerForm = get(".msger-inputarea");
    const msgerInput = get(".msger-input");
    const msgerChat = get(".msger-chat");


    // Icons made by Freepik from www.flaticon.com
    const BOT_IMG = "https://image.flaticon.com/icons/svg/327/327779.svg";
    const PERSON_IMG = "https://image.flaticon.com/icons/svg/145/145867.svg";
    const BOT_NAME = "    ChatBot";
    const PERSON_NAME = "You";

    msgerForm.addEventListener("submit", event => {
      event.preventDefault();

      const msgText = msgerInput.value;
      if (!msgText) return;

      appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
      msgerInput.value = "";
      botResponse(msgText);
    });

    function appendMessage(name, img, side, text) {
      //   Simple solution for small apps
      const msgHTML = `
<div class="msg ${side}-msg">
  <div class="msg-img" style="background-image: url(${img})"></div>
  <div class="msg-bubble">
    <div class="msg-info">
      <div class="msg-info-name">${name}</div>
      <div class="msg-info-time">${formatDate(new Date())}</div>
    </div>
    <div class="msg-text">${text}</div>
  </div>
</div>
`;

      msgerChat.insertAdjacentHTML("beforeend", msgHTML);
      msgerChat.scrollTop += 500;
    }

    function botResponse(rawText) {

      // Bot Response
      $.get("/get", { msg: rawText }).done(function (data) {
        console.log(rawText);
        console.log(data);
        window.speechSynthesis.speak(new SpeechSynthesisUtterance(data));
        const msgText = data;
        appendMessage(BOT_NAME, BOT_IMG, "left", msgText);

      });

    }


    // Utils
    function get(selector, root = document) {
      return root.querySelector(selector);
    }
    function formatDate(date) {
      const h = "0" + date.getHours();
      const m = "0" + date.getMinutes();

      return `${h.slice(-2)}:${m.slice(-2)}`;
    }



  </script>

</body>

</html>
```

CHAPTER-4 Testing

Create training and testing data :

We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as ‘model.h5’.

Testing the bot is a part of our interactive learning process. We keep checking all the possible scenarios initially thought for the bot and train it accordingly

4.1 Test Case 1

User : Hello

HealthBot : Hi there, how can I help?

User : What help you provide?

HealthBot : I can guide you through Adverse drug reaction list, Blood pressure tracking, Hospitals and Pharmacies

4.2 Test Case 2

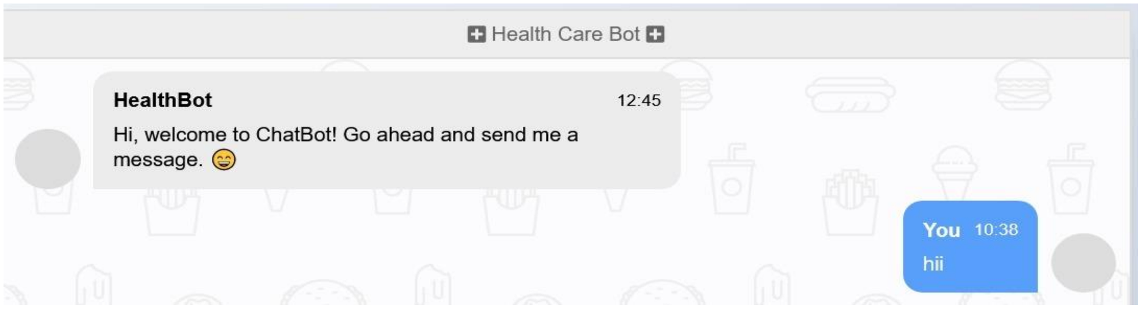
User : thanks

HealthBot : Happy to help!

User : Lookup for hospital

HealthBot : Please provide hospital name or location

CHAPTER-5 Result



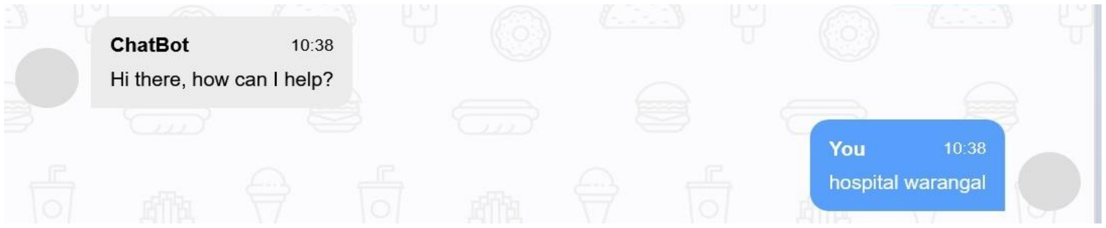
Health Care Bot

HealthBot 12:45

Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 10:38

hii

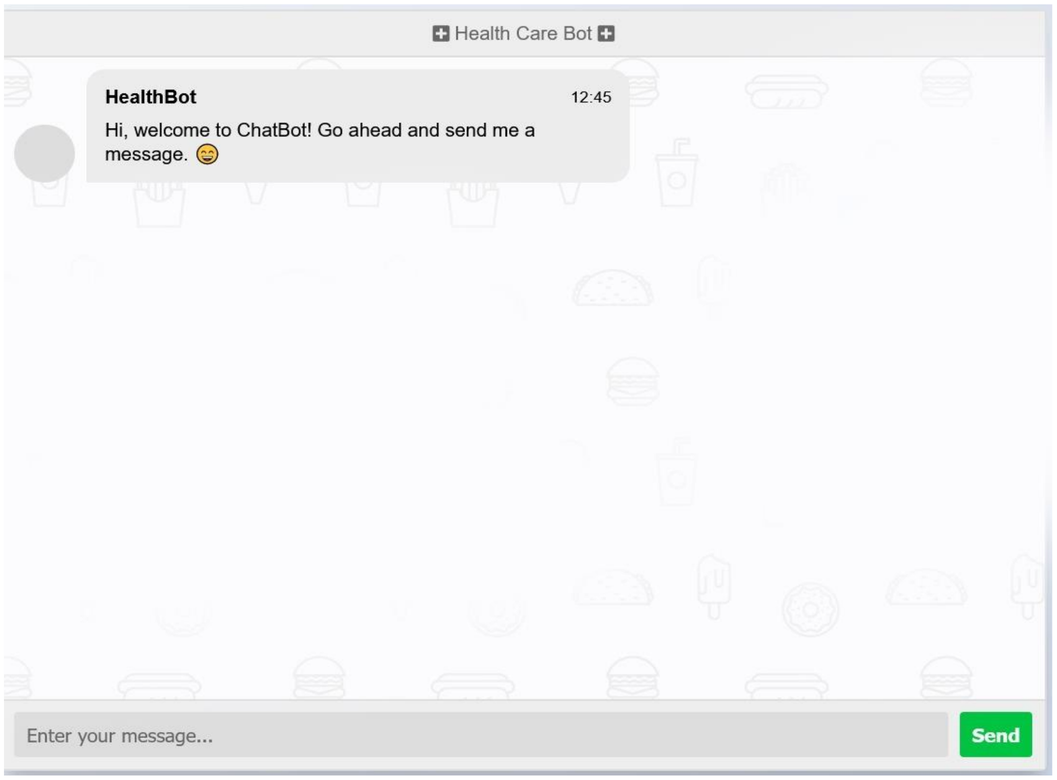


ChatBot 10:38

Hi there, how can I help?

You 10:38

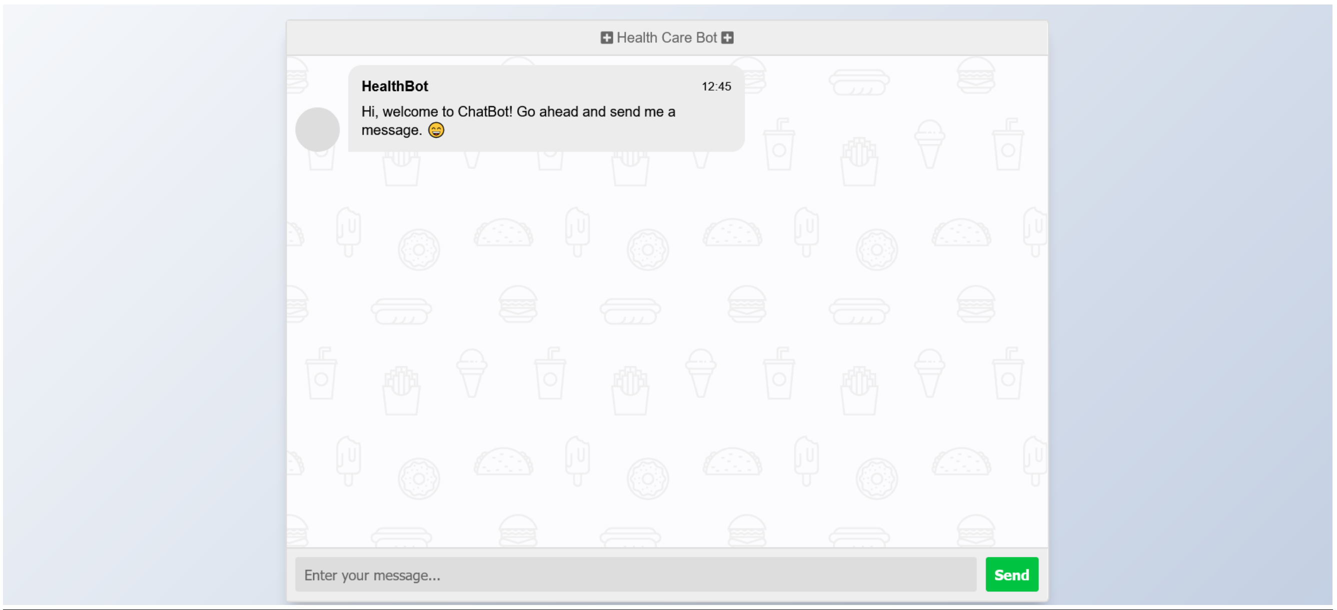
hospital warangal



Health Care Bot

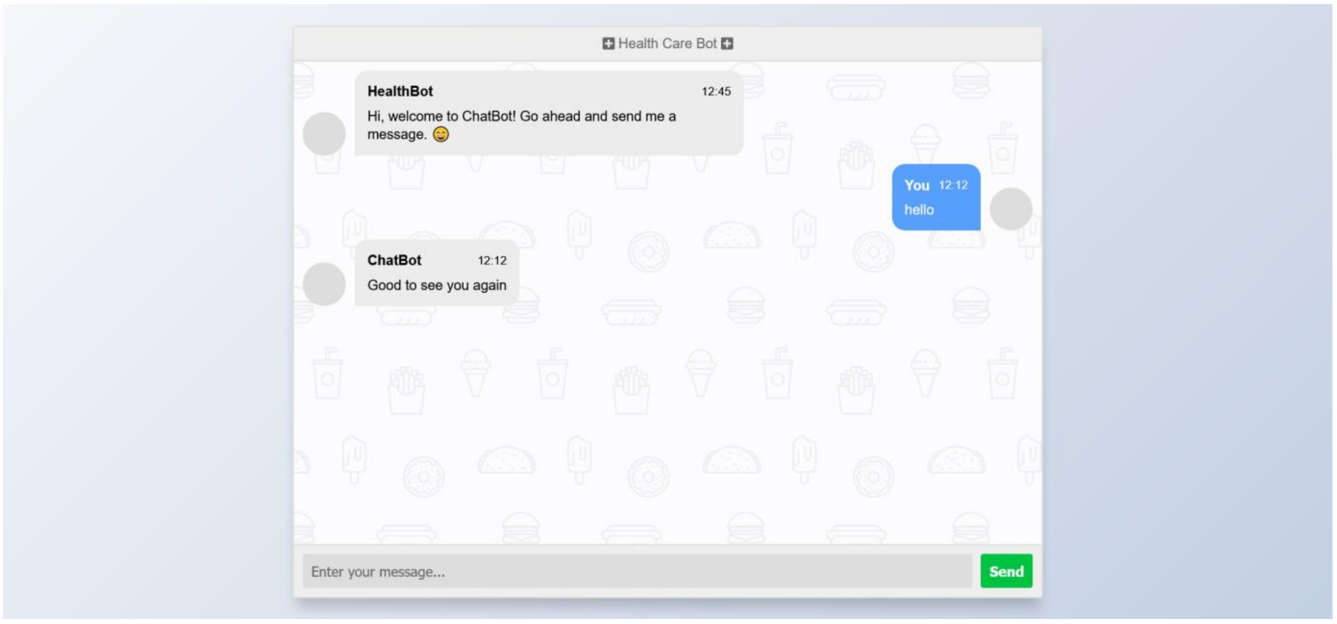
HealthBot 12:45

Hi, welcome to ChatBot! Go ahead and send me a message. 😄



Health Care Bot

HealthBot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

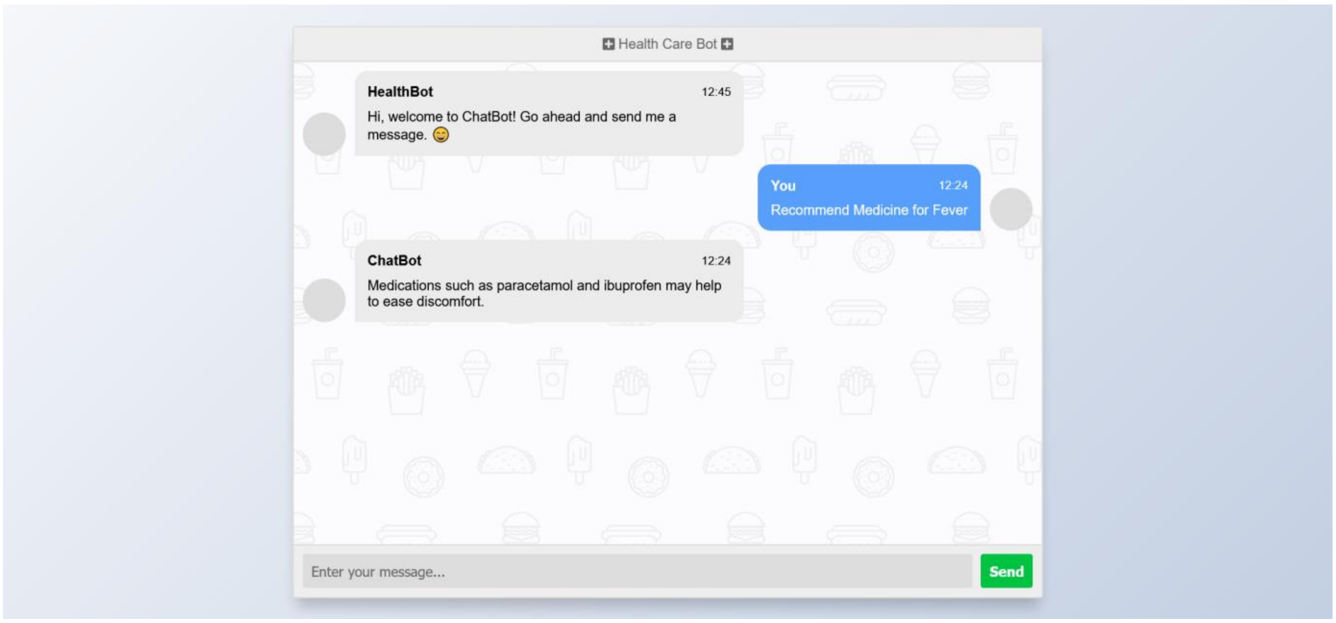


Health Care Bot

HealthBot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 12:12
hello

ChatBot 12:12
Good to see you again

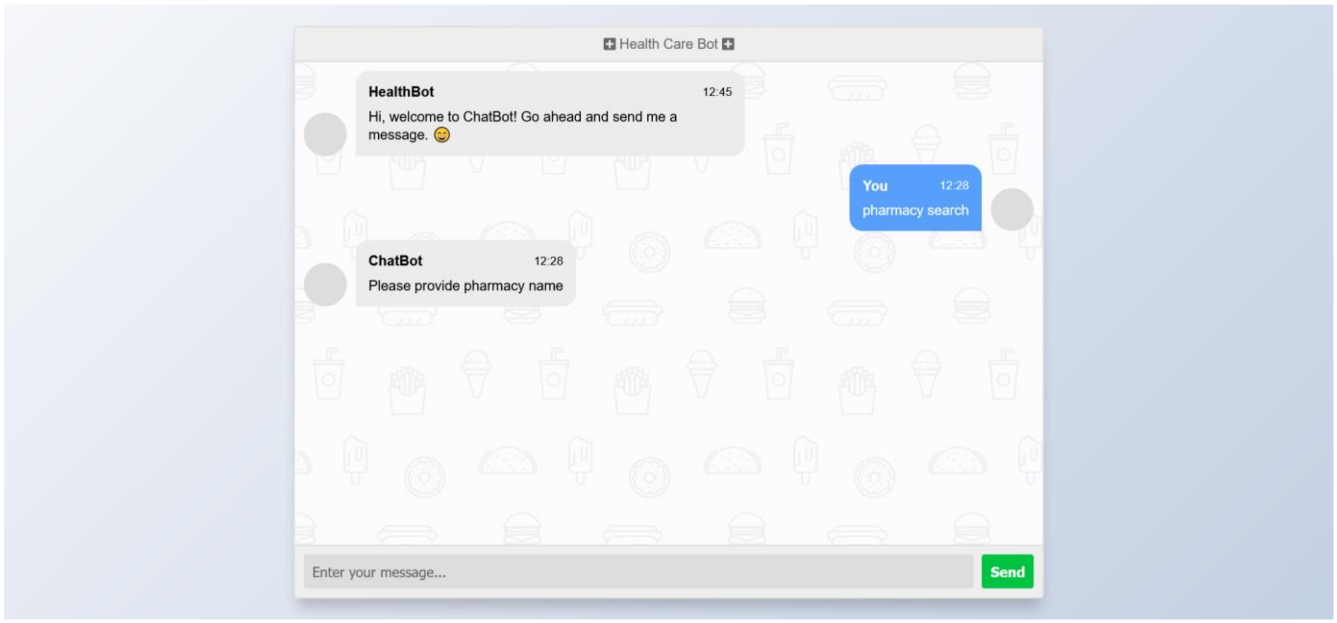


Health Care Bot

HealthBot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 12:24
Recommend Medicine for Fever

ChatBot 12:24
Medications such as paracetamol and ibuprofen may help to ease discomfort.

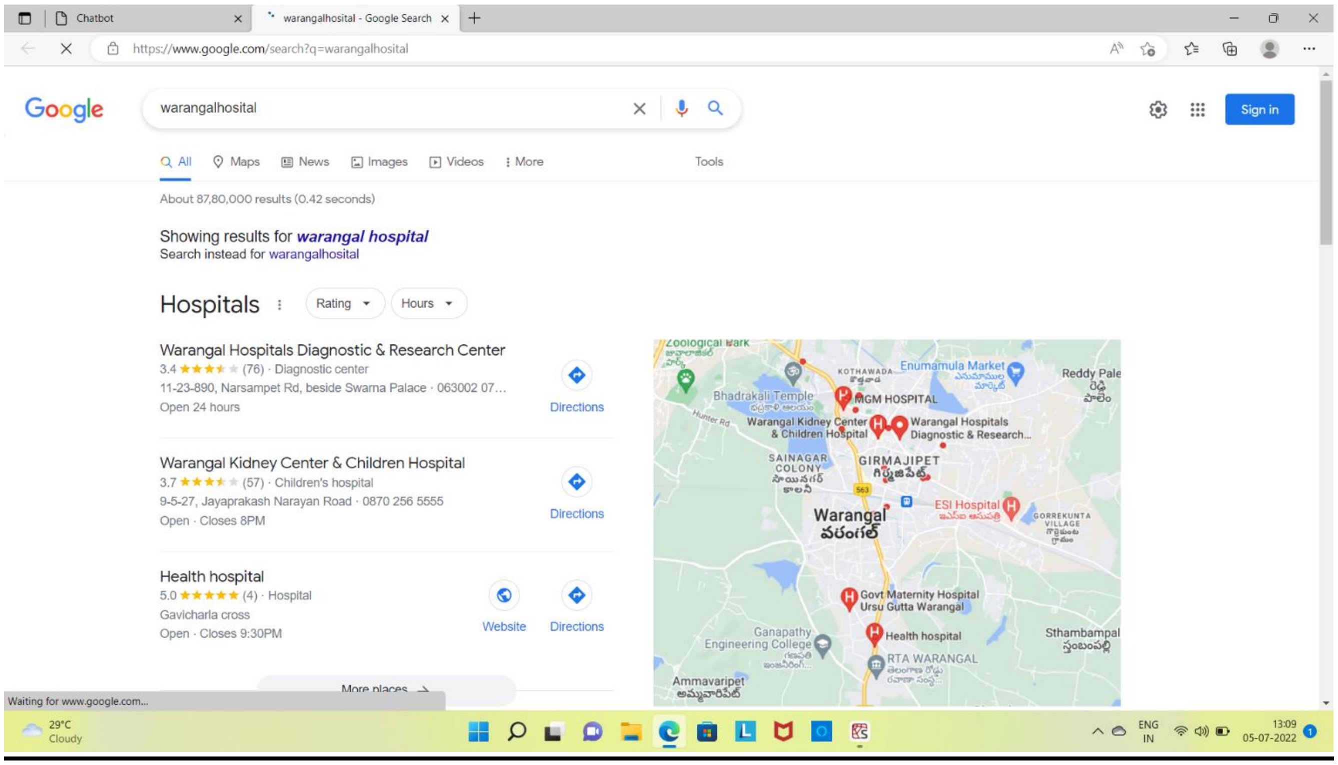


Health Care Bot

HealthBot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 12:28
pharmacy search

ChatBot 12:28
Please provide pharmacy name


Chatbot
warangalhosital - Google Search
https://www.google.com/search?q=warangalhosital
warangalhosital
All
Maps
News
Images
Videos
More
Tools
Sign in

About 87,80,000 results (0.42 seconds)

Showing results for *warangal hospital*
Search instead for warangalhosital

Hospitals

Rating · Hours

Warangal Hospitals Diagnostic & Research Center
3.4 ★★★★ (76) · Diagnostic center
11-23-890, Narsampet Rd, beside Swarna Palace · 063002 07…
Open 24 hours
Directions

Warangal Kidney Center & Children Hospital
3.7 ★★★★ (57) · Children's hospital
9-5-27, Jayaprakash Narayan Road · 0870 256 5555
Open · Closes 8PM
Directions

Health hospital
5.0 ★★★★★ (4) · Hospital
Gavicharla cross
Open · Closes 9:30PM
Website · Directions

More places →

CHAPTER-6 CONCLUSION AND DISCUSSION

6.1 CONCLUSION : The goal of the project is to reduce man-power and to respond to user query at faster rate. Early days, the user's use to send a query mail to the particular site administrator and it would take few days for the site administrator to reply to the mails. Chatbots can overcome this delay, chatbot satisfies the user request or query immediately with relevant responses. These days many websites of banks, educational institutions, business sectors have developed their chatbots to satisfy user request in a faster time. Chatbots are user-friendly artificial .

This paper presents the overall development and working of the chatbot.

6.2 Future Scope

This project can be developed even more by adding multi languages, speech recognition. We can add many more tags to the data set, as the website gets developed. The chat history of a particular user can be sent as a mail to him/her after the conversation is over. This can be done by authorizing the users and receiving their mail id's. This project is a small initiative to make the website user-friendly and easily understandable by the user

Bibliography

1. ALICE. 2002. A.L.I.C.E AI Foundation, http://www.alicebot.org/
2. Kumar, M Naveen, PC Linga Chandar, A Venkatesh Prasad, and K Sumangali (2016). "Android based educational Chatbot for visually impaired people". In: International Conference on Computational Intelligence and Computing Research (ICCIC), 2016 IEEE. IEEE, pp. 1–4. [9]
3. Ranoliya, Bhavika R, Nidhi Raghuwanshi, and Sanjay Singh (2017). "Chatbot for University Related FAQs". In: 2017
4. Mikic, Fernando A, Juan C Burguillo, Mart´ın Llamas, Daniel A Rodr´ıguez, and Eduardo Rodr´ıguez (2009).
5. Emanuela Haller and Traian Rebedea, "Designing a Chatbot that Simulates an Historical Figure", IEEE Conference Publications, July 2013
6. N. Thomas, "An e-business chatbot using aiml and lsa," in Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on. IEEE, 2016,