

# **HEALTH CARE CHAT BOT**

## **A MINI PROJECT REPORT**

**18CSC305J - ARTIFICIAL INTELLIGENCE**

*Submitted by*

**B SIVA CHANIKYA REDDY[RA2111003011738]**

**KILARI ESWAR [RA2111003011755]**

**MODUKURU KARTHIK [RA2111003011756]**

*Under the guidance of*

**Mrs. M RANJANI**

Assistant Professor, Department of Computer Science and Engineering

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Chengalpattu District

**MAY 2023**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that the Mini project report titled "HEALTH CARE CHAT BOT" is the bona fide work of **B SHIVA (HANIKYA REDDY)** [RA2111003011738] **KILARI ESWAR** [RA2111003011755] **MODUKURU KARTHIK** [RA2111003011756] who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

M. Ranjani  
08/02/24

### SIGNATURE

Mrs. M RANJANI  
Assistant Professor  
Department of Computing  
Technologies



M. Pushpalatha

### SIGNATURE

Dr. M. Pushpalatha  
**HEAD OF THE DEPARTMENT**  
Professor & Head  
Department of Computing  
Technologies

## **ABSTRACT**

The Health Care Chatbot AI project aims to develop an intelligent conversational agent that can provide healthcare information and support to users through natural language interactions. The chatbot will utilize machine learning algorithms and NLP techniques to understand user inputs and provide relevant and accurate responses. The system will be designed to handle a wide range of healthcare-related queries, including symptoms, treatments, medications, and medical conditions. The project will leverage existing healthcare datasets to improve the chatbot's accuracy and relevance, while also prioritizing user privacy and data security. Ultimately, the Health Care Chatbot AI project aims to provide users with an accessible and convenient means of accessing reliable healthcare information and support.

# TABLE OF CONTENTS

**ABSTRACT**

**TABLE OF CONTENTS**

**ABBREVIATIONS**

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>2</b>
<b>3</b>	<b>SYSTEM ARCHITECTURE AND DESIGN</b>	<b>3</b>
3.1	Architecture diagram	8
3.2	Description of Module and components	9
<b>4</b>	<b>METHODOLOGY</b>	<b>4</b>
4.1	Methodological Steps	10
<b>5</b>	<b>CODING AND TESTING</b>	<b>5</b>
<b>6</b>	<b>SCREENSHOTS AND RESULTS</b>	<b>11</b>
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>12</b>
7.1	Conclusion	
7.2	Future Enhancement	
	<b>REFERENCES</b>	<b>13</b>

# **CHAPTER 1**

## **INTRODUCTION**

A computer programme created for the healthcare sector's chat bot initiative simulates talks with real consumers. The chat bot intends to help patients and healthcare professionals by giving medical advice, responding to inquiries about symptoms and treatments, directing users to resources for healthcare, and helping with appointment scheduling. The need for rapid and easy access to medical information as well as the rising demand for healthcare services have increased the significance of the health chat bot project. Users can communicate with the healthcare system through chat bots, which provide an easy-to-use and accessible method of communication. The health care chat bot project makes use of a number of technologies, including artificial intelligence, machine learning, and natural language processing, to comprehend and reply to user requests. By analysing user inputs and delivering pertinent information, the chat bot can provide precise and prompt responses. Overall, the health care chat bot project is a useful tool for the healthcare sector, offering a low-cost and effective way to provide consumers and healthcare providers with medical advice and support.

## **CHAPTER 2**

### **LITERATURE SURVEY**

In recent years, the use of chatbots in the healthcare industry has increased. A number of research have been carried out to determine how useful chatbots are in various healthcare settings.

Title of Paper 1: Improving LMS Experience with AIML Base and Retrieval Base R-based chatbot Language

Brief Description: The usage of several techniques, including N-gram, Stemming, TF-IDF, and cosine similarity, is covered in detail in the study. This paper explains how to apply these techniques to produce an optimised outcome rapidly. It details various inquiries and how a chatbot might respond to them. For convenience, they also specified the functional architecture of databases and offered test cases for algorithms. The essence of chatbot quality

Title of Paper 2: Artificial Intelligence Based Personal Assistant, Brief

Description:

Here In this work, a chatbot that acts as a user's personal healthcare assistant is built for healthcare- related purposes. A user can interact with a medical assistant via a dialogue interface. It offers features like ailment diagnosis based on symptoms reported by the user, definitions of medical terms, advice from doctors, scheduling of treatments, and tracking and monitoring of the user's health metrics. Both offline and online performance were looked at overall. To assess the effectiveness and quality of the system, they also ran a number of tests.

To create an eHealth environment for patients' convenience, numerous studies on this topic have been done in the recent years.

Title of Paper 3: Human-to-Machine Conversation Modelling

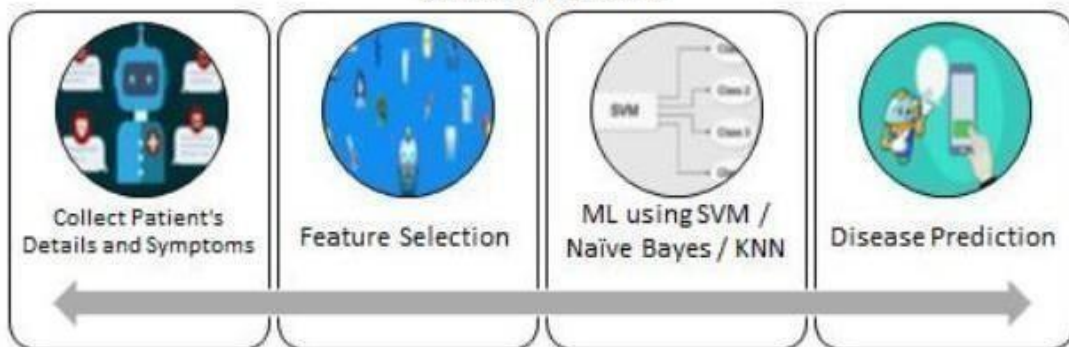
Brief Description: In this study, hardware and software were combined to develop a chatbot system. This chatbot is Bluetooth-enabled and can move. They controlled chatbots through voice communication. A database that is kept on the Raspberry Pi will be searched using the user's query. This chatbot only functions with particular illnesses like the common cold, typhoid, malaria, etc.

# CHAPTER 3

## SYSTEM ARCHITECTURE AND DESIGN



**Medical Chabot**



This is the proposed architecture for our AI-Powered Health Chatbot, and we will give further explain for every component in the next upcoming papers. First of all, the end user interacts with the chatbot through a client platform. It's important to be user friendly and give an excellent UX.

Each time, the user is having a request, it's routed to the NLP Engine using the appropriate API's, in the NLP Engine, and with NLU, the chatbot understands the request and format the data into understandable form that can be understood by the Core Engine. Once the Core Engine receives the formatted data, it searches using Deep Learning Algorithms, the appropriate response and send it back to the NLP Engine.

## **CHAPTER 4**

### **METHODOLOGY**

The following steps could be used in the technique for creating a health care chatbot:

A health care chatbot's target audience must be determined before any other steps can be taken to develop it. The chatbot might be created to meet the needs of patients, physicians, nurses, or other healthcare providers. Designing the chatbot's functionalities would be aided by an understanding of the needs of the target audience.

**Identify the chatbot's objectives:** The next stage is to identify the objectives of the chatbot. The objectives should match the requirements of the target audience. For instance, a chatbot created just for patients would be intended to answer general health-related concerns, arrange appointments, and send out reminders.

**Gather information and create a database:** A substantial amount of data that can aid the chatbot in comprehending and responding to user inquiries is necessary in order to construct one. The information might be found in patient records, clinical data, and medical literature. By using this information to teach the chatbot, a knowledge base can be created.

**Select the chatbot development platform and tools:** There are a variety of chatbot development platforms and tools available. The objectives and needs of the chatbot will determine the platform and technologies to use.

**Create the dialogue flow for the chatbot:** This will establish how the chatbot will communicate with the user. The discussion flow has to be planned to guarantee that the user's inquiries are answered truthfully and effectively. **Create the natural language processing (NLP) system for the chatbot:** The chatbot's capacity to comprehend natural language is a crucial feature. The NLP system needs to be able to handle many dialects, accents, and languages.

**Test and improve the chatbot:** After the chatbot is created, it should be put to the test to make sure it works as intended. Any problems or flaws with the chatbot can be found by conducting user testing. The chatbot can be enhanced and developed based on the input.

**Deploy the chatbot:** After it has been tested and improved, the chatbot can be made available to the intended audience. To make sure the chatbot is operating properly and answering users' questions, regular checks should be made.



# CHAPTER 5

## CODING AND TESTING

```

1 ~ import re
2 import pandas as pd
3 import pytsx3
4 from sklearn import preprocessing
5 from sklearn.tree import DecisionTreeClassifier, _tree
6 import numpy as np
7 from sklearn.model_selection import train_test_split
8 from sklearn.model_selection import cross_val_score
9 from sklearn.svm import SVC
10 import csv
11 import warnings
12 warnings.filterwarnings("ignore", category=DeprecationWarning)
13
14
15 training = pd.read_csv('Data/Training.csv')
16 testings = pd.read_csv('Data/Testing.csv')
17 cols = training.columns
18 cols = cols[:-1]
19 x = training[cols]
20 y = training['prognosis']
21 y1 = y
22
23
24 reduced_data = training.groupby(training['prognosis']).max()
25
26 #mapping strings to numbers
27 le = preprocessing.LabelEncoder()
28 le.fit(y)
29 y = le.transform(y)

```

```

32 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
33 testx = testing[cols]
34 testy = testing['prognosis']
35 testy = le.transform(testy)
36 clf1 = DecisionTreeClassifier()
37 clf = clf1.fit(x_train, y_train)
38 # print(clf.score(x_train, y_train))
39 # print ("cross result=====")
40 scores = cross_val_score(clf, x_test, y_test, cv=3)
41 # print (scores)
42 print (scores.mean())
43 model=SVC()
44 model.fit(x_train, y_train)
45 print("for svm: ")
46 print(model.score(x_test, y_test))
47 importances = clf.feature_importances_
48 indices = np.argsort(importances)[::-1]
49 features = cols
50
51 def read(nstr):
52     engine = pytsx3.init()
53     engine.setProperty('voice', "english+f5")
54     engine.setProperty('rate', 150)
55     engine.say(nstr)
56     engine.runAndWait()
57     engine.stop()
58
59 severityDictionary=dict()
60 description_list = dict()
61 precautionDictionary=dict()
62 symptoms_dict = {}

```

```

64
65 for index, symptom in enumerate(x):
66     symptoms_dict[symptom] = index
67
68 1 usage
69
70 def calc_condition(exp,days):
71     sum=0
72     for item in exp:
73         sum=sum+severityDictionary[item]
74     if((sum+days)/(len(exp)+1)>13):
75         print("You should take the consultation from doctor. ")
76     else:
77         print("It might not be that bad but you should take precautions.")
78
79
80 1 usage
81
82 def getDescription():
83     global description_list
84     with open('MasterData/symptom_Description.csv') as csv_file:
85         csv_reader = csv.reader(csv_file, delimiter=',')
86         line_count = 0
87         for row in csv_reader:
88             _description={row[0]:row[1]}
89             description_list.update(_description)
90
91 1 usage
92
93 def getSeverityDict():
94     global severityDictionary
95     with open('MasterData/symptom_severity.csv') as csv_file:
96         csv_reader = csv.reader(csv_file, delimiter=',')
97         line_count = 0

```

```

98
99     try:
100         for row in csv_reader:
101             _diction={row[0]:int(row[1])}
102             severityDictionary.update(_diction)
103     except:
104         pass
105
106 1 usage
107
108 def getprecautionDict():
109     global precautionDictionary
110     with open('MasterData/symptom_precaution.csv') as csv_file:
111
112         csv_reader = csv.reader(csv_file, delimiter=',')
113         line_count = 0
114         for row in csv_reader:
115             _prec={row[0]:[row[1],row[2],row[3],row[4]]}
116             precautionDictionary.update(_prec)
117
118 1 usage
119
120 def getInfo():
121     print("-----HealthCare ChatBot-----")
122     print("\nYour Name? \t\t\t\t\t,ends*->")
123     name=input("")
124     print("Hello, ",name)
125
126 1 usage
127
128 def check_pattern(dis_list,inp):

```

```

113 def check_pattern(dis_list,inp):
114     pred_list=[]
115     inp=inp.replace(' ','_')
116     patt = f"^{inp}$"
117     regexp = re.compile(patt)
118     pred_list=[item for item in dis_list if regexp.search(item)]
119     if(len(pred_list)>0):
120         return 1,pred_list
121     else:
122         return 0,[]
123
124 usage
125 def sec_predict(symptoms_exp):
126     df = pd.read_csv('Data/Training.csv')
127     X = df.iloc[:, :-1]
128     y = df['prognosis']
129     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=28)
130     rf_clf = DecisionTreeClassifier()
131     rf_clf.fit(X_train, y_train)
132
133     symptoms_dict = {symptom: index for index, symptom in enumerate(X)}
134     input_vector = np.zeros(len(symptoms_dict))
135     for item in symptoms_exp:
136         input_vector[[symptoms_dict[item]]] = 1
137
138     return rf_clf.predict([input_vector])
139
140 usage
141 def print_disease(node):
142     node = node[0]
143     val = node.nonzero()
144     disease = to.inverse_transform(val[0])
145     return list(map(lambda x:x.strip(),list(disease)))
146
147 usage
148 def tree_to_code(tree, feature_names):
149     tree_ = tree.tree_
150     feature_name = [
151         feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
152         for i in tree_.feature
153     ]
154
155     chk_dis=".".join(feature_names).split(",")
156     symptoms_present = []
157
158     while True:
159
160         print("\nEnter the symptom you are experiencing \t\t",end=">")
161         disease_input = input("")
162         conf,cnf_dis=check_pattern(chk_dis,disease_input)
163         if conf==1:
164             print("searches related to input: ")
165             for num,it in enumerate(cnf_dis):
166                 print(num,"-",it)
167             if num!=0:
168                 print(f"Select the one you meant (0 - {num}): ", end="")
169                 conf_inp = int(input(""))
170             else:

```

```

168         conf_inp=0
169
170         disease_input=cnf_dis[conf_inp]
171         break
172         # print("Did you mean: ",cnf_dis,"?(yes/no) :",end="")
173         # conf_inp = input("")
174         # if(conf_inp=="yes"):
175         #     break
176     else:
177         print("Enter valid symptom.")
178
179     while True:
180         try:
181             num_days=int(input("Okay. From how many days ? : "))
182             break
183         except:
184             print("Enter valid input.")
185     def recurse(node, depth):
186         indent = " " * depth
187         if tree_.feature[node] != _tree.TREE_UNDEFINED:
188             name = feature_name[node]
189             threshold = tree_.threshold[node]
190
191             if name == disease_input:
192                 val = 1
193             else:
194                 val = 0
195             if val <= threshold:
196                 recurse(tree_.children_left[node], depth + 1)

```

```

198         symptoms_present.append(name)
199         recurse(tree_.children_right[node], depth + 1)
200     else:
201         present_disease = print_disease(tree_.value[node])
202         # print( "You may have " + present_disease )
203         red_cols = reduced_data.columns
204         symptoms_given = red_cols[reduced_data.loc[present_disease].values[0].nonzero()]
205         # dis_list=list(symptoms_present)
206         # if len(dis_list)!=0:
207             # print("symptoms present " + str(list(symptoms_present)))
208             # print("symptoms given " + str(list(symptoms_given)) )
209             print("Are you experiencing any ")
210             symptoms_exp=[]
211             for syms in list(symptoms_given):
212                 inp=""
213                 print(syms,"? : ",end="")
214                 while True:
215                     inp=input("")
216                     if(inp=="yes" or inp=="no"):
217                         break
218                     else:
219                         print("provide proper answers i.e. (yes/no) : ",end="")
220                 if(inp=="yes"):
221                     symptoms_exp.append(syms)
222
223             second_prediction=sec_predict(symptoms_exp)
224             # print(second_prediction)
225             calc_condition(symptoms_exp,num_days)
226             if(present_disease[0]==second_prediction[0]):
227                 print(f"you may have {present_disease[0]}")
228             else:
229                 print(f"you may have {second_prediction[0]}")
230
231             getDescription() : with open('MasterData/symptom_D_') for row in csv.reader

```

```

M+ README.md chat_bot.py requirements.txt symptom_Description.csv
224
225     if(present_disease[0]==second_prediction[0]):
226         print("You may have ", present_disease[0])
227         print(description_list[present_disease[0]])
228
229         # readn(f"You may have {present_disease[0]}")
230         # readn(f"{description_list[present_disease[0]]}")
231
232     else:
233         print("You may have ", present_disease[0], "or ", second_prediction[0])
234         print(description_list[present_disease[0]])
235         print(description_list[second_prediction[0]])
236
237         # print(description_list[present_disease[0]])
238         precaution_list=precautionDictionary[present_disease[0]]
239         print("Take following measures : ")
240         for i,j in enumerate(precaution_list):
241             print(i+1,",",j)
242
243         # confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
244         # print("confidence level is " + str(confidence_level))
245
246     recurse(0, 1)
247     getSeverityDict()
248     getDescription()
249     getprecautionDict()
250     getInfo()
251     tree_to_code(clf,cols)
252     print("-----")
253

```

Code:

```

import re
import pandas as pd
import pytsx3 from
sklearn import preprocessing from sklearn.tree
import DecisionTreeClassifier,_tree import numpy as
np from sklearn.model_selection import
train_test_split from sklearn.model_selection import
cross_val_score from sklearn.svm import SVC
import csv import
warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

```

```

training = pd.read_csv('Data/Training.csv')
testing = pd.read_csv('Data/Testing.csv')
cols = training.columns cols = cols[:-1] x =
training[cols] y = training['prognosis']
y1 = y

```

```

reduced_data = training.groupby(training['prognosis']).max()

```

```

#mapping strings to numbers
le = preprocessing.LabelEncoder()
le.fit(y)
y = le.transform(y)

```

```

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
testx = testing[cols] testy = testing['prognosis'] testy = le.transform(testy)

```

```

clf1 = DecisionTreeClassifier()
clf = clf1.fit(x_train,y_train) #
print(clf.score(x_train,y_train)) #
print ("cross result=====")
scores = cross_val_score(clf, x_test, y_test, cv=3)
# print (scores) print
(scores.mean())

```

```

model=SVC()
model.fit(x_train,y_train) print("for
svm: ")
print(model.score(x_test,y_test))

importances = clf.feature_importances_ indices
= np.argsort(importances)[::-1] features = cols

def readn(nstr):
    engine = pyttsx3.init()

    engine.setProperty('voice', "english+f5")
    engine.setProperty('rate', 130)

    engine.say(nstr)
    engine.runAndWait() engine.stop()

severityDictionary=dict() description_list
= dict()
precautionDictionary=dict()

symptoms_dict = {}

for index, symptom in enumerate(x):
    symptoms_dict[symptom] = index def
    calc_condition(exp,days):    sum=0
    for item in exp:
        sum=sum+severityDictionary[item]
    if((sum*days)/(len(exp)+1)>13):    print("You should
    take the consultation from doctor. ")    else:
        print("It might not be that bad but you should take precautions.")

def getDescription():    global description_list    with
    open('MasterData/symptom_Description.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0    for row in csv_reader:
            _description={row[0]:row[1]}
            description_list.update(_description)

def getSeverityDict():
    global severityDictionary    with
    open('MasterData/symptom_severity.csv') as csv_file:

        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0    try:    for row in
        csv_reader:
            _diction={row[0]:int(row[1])}
            severityDictionary.update(_diction)    except:
            pass

def getprecautionDict():
    global precautionDictionary    with
    open('MasterData/symptom_precaution.csv') as csv_file:

        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0    for row in csv_reader:

```

```

        _prec={row[0]:[row[1],row[2],row[3],row[4]]}
precautionDictionary.update(_prec)

def getInfo():    print(".....HealthCare ChatBot.....")
    print("\nYour Name? \t\t\t",end="->")    name=input("")    print("Hello, ",name)

def check_pattern(dis_list,inp):
    pred_list=[]    inp=inp.replace(' ','_')    patt = f"{inp}"
    regexp = re.compile(patt)    pred_list=[item for item in dis_list if regexp.search(item)]    if(len(pred_list)>0):
    return 1,pred_list    else:
        return 0,[]
def sec_predict(symptoms_exp):
    df = pd.read_csv('Data/Training.csv')
    X = df.iloc[:, :-1]    y = df['prognosis']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=20)
    rf_clf = DecisionTreeClassifier()
    rf_clf.fit(X_train, y_train)

    symptoms_dict = {symptom: index for index, symptom in enumerate(X)}
    input_vector = np.zeros(len(symptoms_dict))
    for item in symptoms_exp:
        input_vector[symptoms_dict[item]] = 1

    return rf_clf.predict([input_vector])

def print_disease(node):    node = node[0]    val
    = node.nonzero()    disease =
    le.inverse_transform(val[0])    return
    list(map(lambda x:x.strip(),list(disease)))

def tree_to_code(tree, feature_names):
    tree_ = tree.tree_
    feature_name = [
        feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
        for i in tree_.feature
    ]

    chk_dis=",".join(feature_names).split(",")
    symptoms_present = []

    while True:

        print("\nEnter the symptom you are experiencing \t\t",end="->")
        disease_input = input("")
        cnf,cnf_dis=check_pattern(chk_dis,disease_input)
        if cnf==1:    print("searches related to input: ")
        for num,it in enumerate(cnf_dis):
            print(num,")",it)
        if num!=0:
            print(f"Select the one you meant (0 - {num}): ", end="")
            conf_inp = int(input(""))
        else:
            conf_inp=0

        disease_input=cnf_dis[conf_inp]
        break
        # print("Did you mean: ",cnf_dis,"?(yes/no) :",end="")
        # conf_inp = input("")    # if(conf_inp=="yes"):
        #     break
    else:
        print("Enter valid symptom.")

    while True:
try:

```

```

        num_days=int(input("Okay. From how many days ? : "))
    break    except:        print("Enter valid input.")    def
recurse(node, depth):        indent = " " * depth        if
tree_.feature[node] != _tree.TREE_UNDEFINED:
        name = feature_name[node]
        threshold = tree_.threshold[node]

        if name == disease_input:
            val = 1        else:
                val = 0        if val <= threshold:
                    recurse(tree_.children_left[node], depth + 1)
                else:
                    symptoms_present.append(name)
                    recurse(tree_.children_right[node], depth + 1)        else:
                        present_disease = print_disease(tree_.value[node])
                        # print("You may have " + present_disease )
                        red_cols = reduced_data.columns
                        symptoms_given = red_cols[reduced_data.loc[present_disease].values[0].nonzero()]        # dis_list=list(symptoms_present)        # if
                        len(dis_list)!=0:
                            # print("symptoms present " + str(list(symptoms_present)))
                            # print("symptoms given " + str(list(symptoms_given)) )
                            print("Are you experiencing any ")        symptoms_exp=[]
                            for syms in list(symptoms_given):        inp=""
                            print(syms,"? : ",end="")        while True:
                                inp=input("")        if(inp=="yes" or inp=="no"):
                                    break        else:
                                        print("provide proper answers i.e. (yes/no) : ",end="")
                                if(inp=="yes"):
                                    symptoms_exp.append(syms)

                                second_prediction=sec_predict(symptoms_exp)
                                # print(second_prediction)
                                calc_condition(symptoms_exp,num_days)
                                if(present_disease[0]==second_prediction[0]):
                                    print("You may have ", present_disease[0])
                                    print(description_list[present_disease[0]])

                                # readn(f"You may have {present_disease[0]}")
                                # readn(f"{description_list[present_disease[0]]}")

                            else:
                                print("You may have ", present_disease[0], "or ", second_prediction[0])
                                print(description_list[present_disease[0]])
                                print(description_list[second_prediction[0]])

                                # print(description_list[present_disease[0]])
                                precaution_list=precautionDictionary[present_disease[0]]
                                print("Take following measures : ")        for i,j in
                                enumerate(precaution_list):
                                    print(i+1,"")j)

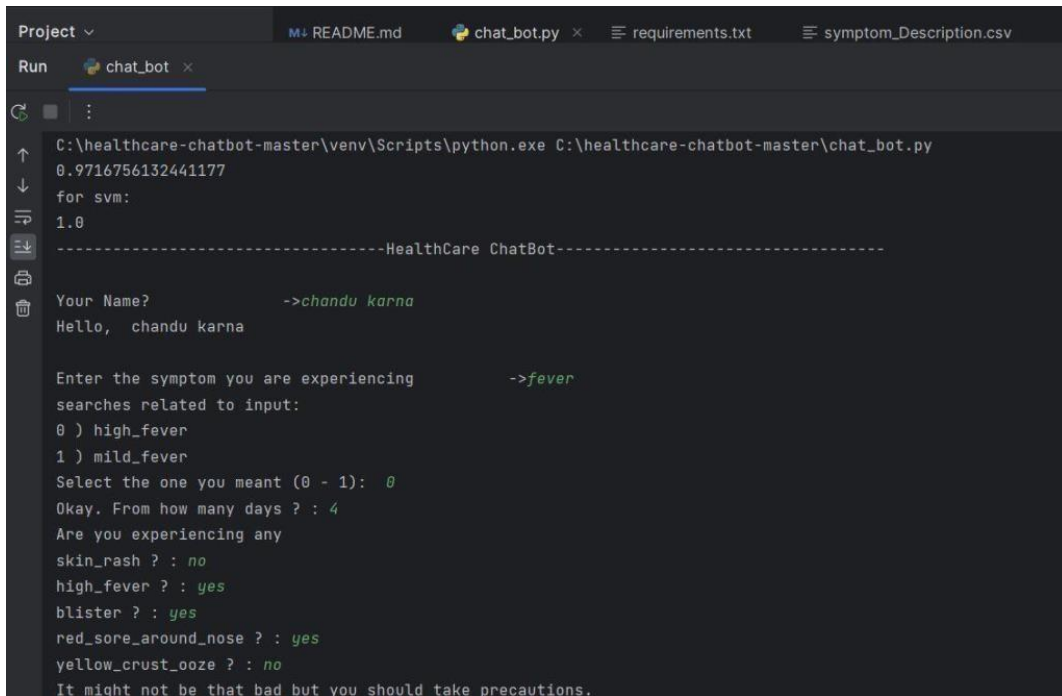
                                # confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
                                # print("confidence level is " + str(confidence_level))

                                recurse(0, 1)
                                getSeverityDict()
                                getDescription()
                                getprecautionDict() getInfo()
                                tree_to_code(clf,cols)
                                print(".....")

```

## CHAPTER 6

### SCREENSHOTS AND RESULTS:



The screenshot shows a terminal window with the following output:

```
C:\healthcare-chatbot-master\venv\Scripts\python.exe C:\healthcare-chatbot-master\chat_bot.py
0.9716756132441177
for svm:
1.0
-----HealthCare ChatBot-----

Your Name?          ->chandu karna
Hello,  chandu karna

Enter the symptom you are experiencing          ->fever
searches related to input:
0 ) high_fever
1 ) mild_fever
Select the one you meant (0 - 1):  0
Okay. From how many days ? : 4
Are you experiencing any
skin_rash ? : no
high_fever ? : yes
blister ? : yes
red_sore_around_nose ? : yes
yellow_crust_ooze ? : no
It might not be that bad but you should take precautions.
```

## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

In conclusion,

The suggested solution helps hospitals and medical centres by enabling patients to ask questions about their health freely via voice or text requests.

The machine gathers data from medical API diagnostic and speaks out with disease treatments. Comparatively speaking, SVM classification accuracy is higher than KNN and Naive Bayes algorithms.

In comparison to other machine learning algorithms, SVM generates 92.33% accuracy and is thus utilised to accurately predict the disease. It also saves time and space.

Future Improvement

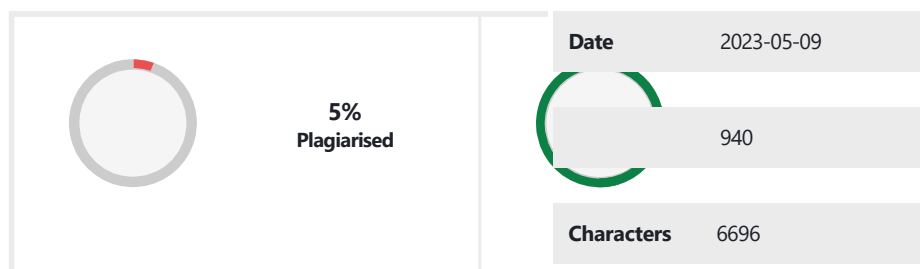
Medical chatbots can be expanded and utilised extensively with other medical systems that allow for prediction by utilising the benefits of the SVM algorithm. It can also be used to plan doctor visits and notify patients of upcoming appointments or checkups. it can also be extended to collect patients' feedback and this will help medical organizations to improve their processes.



## REFERENCES

- [1] Mohammed Javed, P. Nagabhushan, B.B. Chaudhari, “A Direct Approach for Word and Character Segmentation in Run-Length Compressed Documents with an Application to Word Spotting”, 13th International Conference on Document Analysis and Recognition (ICDAR), 2015.
- [2] Naeun Lee, Kirak Kim, Taeseon Yoon, “Implementation of Robot Journalism by Programming Custombot using Tokenization and Custom Tagging”, 2017.
- [3] Tao Jiang, Hongzhi Yu, Yangkyi Jam, “Tibetan Word Segmentation Systems based on Conditional Random Fields”, 2011.
- [4] Jerome r. Bellagarda, “Parts-Of-Speech tagging by Latent Analogy”, IEEE Journal of Selected Topics in Signal Processing, Vol. 4, No. 6, 2010.
- [5] Liner Yang, Meishan Zhang, Yang Liu, Maosong Sun, Nan Yu, Guohong Fu, “Joint POS Tagging and Dependency Parsing with Transition-based Neural Networks”, 2018.

## PLAGIARISM SCAN REPORT



### Content Checked For Plagiarism

A computer programme created for the healthcare sector's chat bot initiative simulates talks with real consumers. The chat bot intends to help patients and healthcare professionals by giving medical advice, responding to inquiries about symptoms and treatments, directing users to resources for healthcare, and helping with appointment scheduling. The need for rapid and easy access to medical information as well as the rising demand for healthcare services have increased the significance of the health chat bot project. Users can communicate with the healthcare system through chat bots, which provide an easy-to-use and accessible method of communication. The health care chat bot project makes use of a number of technologies, including artificial intelligence, machine learning, and natural language processing, to comprehend and reply to user requests. By analysing user inputs and delivering pertinent information, the chatbot can provide precise and prompt responses. Overall, the health care chat bot project is a useful tool for the healthcare sector, offering a low-cost and effective way to provide consumers and healthcare providers with medical advice and support. CHAPTER 2

#### LITERATURE SURVEY

In recent years, the use of chatbots in the healthcare industry has increased. A number of research have been carried out to determine how useful chatbots are in various healthcare settings. Title of Paper 1: Improving LMS Experience with AIML Base and Retrieval Base R-based chatbot Language Brief Description: The usage of several techniques, including N-gram, Stemming, TF-IDF, and cosine similarity, is covered in detail in the study. This paper explains how to apply these techniques to produce an optimised outcome rapidly. It details various inquiries and how a chatbot might respond to them. For convenience, they also specified the functional architecture of databases and offered test cases for algorithms. The essence of chatbot quality

Title of Paper 2: Artificial Intelligence Based Personal Assistant, Brief Description:

Here In this work, a chatbot that acts as a user's personal healthcare assistant is built for healthcare-related purposes. A user can interact with a medical assistant via a dialogue interface. It offers features like ailment diagnosis based on symptoms reported by the user, definitions of medical terms, advice from doctors, scheduling of treatments, and tracking and monitoring of the user's health metrics. Both offline and online performance were looked at overall. To assess the effectiveness and quality of the system, they also ran a number of tests. To create an eHealth environment for patients' convenience, numerous studies on this topic have been done in the recent years.

Title of Paper 3: Human-to-Machine Conversation Modelling Brief Description:

In this study, hardware and software were combined to develop a chatbot system. This chatbot is Bluetooth-enabled and can move. They controlled chatbots through voice communication. A database that is kept on the Raspberry Pi will be searched using the user's query. This chatbot only functions with particular illnesses like the common cold, typhoid, malaria, etc. CHAPTER 3 SYSTEM ARCHITECTURE AND DESIGN This is the proposed architecture for our AI-Powered Health Chatbot, and we will give further explain for every component in the next upcoming papers. First of all, the end

user interacts with the chatbot through a client 14

platform. It's important to be user friendly and give an excellent UX.

Each time, the user is having a request, it's routed to the NLP Engine using the appropriate API's, in the NLP Engine, and with NLU, the chatbot understands the request and format the data into understandable form that can be understood by the Core Engine. Once the Core Engine receives the formatted data, it searches using Deep Learning Algorithms, the appropriate response and send it back to the NLP Engine.

## CHAPTER 4 METHODOLOGY

The following steps could be used in the technique for creating a health care chatbot: A health care chatbot's target audience must be determined before any other steps can be taken to develop it. The chatbot might be created to meet the needs of patients, physicians, nurses, or other healthcare providers. Designing the chatbot's functionalities would be aided by an understanding of the needs of the target audience. Identify the chatbot's objectives: The next stage is to identify the objectives of the chatbot. The objectives should match the requirements of the target audience. For instance, a chatbot created just for patients would be intended to answer general health-related concerns, arrange appointments, and send out reminders.

Gather information and create a database: A substantial amount of data that can aid the chatbot in comprehending and responding to user inquiries is necessary in order to construct one. The information might be found in patient records, clinical data, and medical literature.

By using this information to teach the chatbot, a knowledge base can be created.

Select the chatbot development platform and tools: There are a variety of chatbot development platforms and tools available. The objectives and needs of the chatbot will determine the platform and technologies to use.

Create the dialogue flow for the chatbot: This will establish how the chatbot will communicate with the user. The discussion flow has to be planned to guarantee that the user's inquiries are answered truthfully and effectively.

Create the natural language processing (NLP) system for the chatbot: The chatbot's capacity to comprehend natural language is a crucial feature. The NLP system needs to be able to handle many dialects, accents, and languages.

Test and improve the chatbot: After the chatbot is created, it should be put to the test to make sure it works as intended. Any problems or flaws with the chatbot can be found by conducting user testing. The chatbot can be enhanced and developed based on the input.

Deploy the chatbot: After it has been tested and improved, the chatbot can be made available to the intended audience. To make sure the chatbot is operating properly and answering users' questions, regular checks should be made.

## CHAPTER 5 CODING AND TESTING

### Matched Source

#### Similarity 11%

**Title:** **AI-Powered Health Chatbots: Toward a general architecture** by A Khadija · 2021 · Cited by 7 — First of all, the end user interacts with the chatbot through a client platform. It's important to be user friendly and give an excellent UX.

<https://www.sciencedirect.com/science/article/pii/S1877050921014459/pdf?md5=638dca7503c966ea262d98f99ad06103>