A project report on

HEALTHCARE CHAT-BOT

Submitted in partial fulfillment of the requirements for the Degree of

B. Tech in COMPUTER SCIENCE

by

Priyansh Choudhary (1805589)
Shweta Nathany (1806164)
Debalina Mazumder (1828064)

under the guidance of

PROF. JYOTIPRAKASH MISHRA

School of Computer Engineering
Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar

November 2021

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University U/S 3 of the UGC Act, 1956

CERTIFICATE

This is to certify that the project report entitled “**HEALTHCARE CHAT-BOT**” submitted by

Priyansh Choudhary	**1805589**
Shweta Nathany	**1806164**
Debalina Mazumder	**1828064**

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology** in **COMPUTER SCIENCE Engineering** is a bonafide record of the work carried out under my(our) guidance and supervision at School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

Signature of Supervisor 2 (if applicable)	Signature of Supervisor 1
NAME OF THE SUPERVISOR 2	NAME OF THE SUPERVISOR 1
Academic affiliation	Academic affiliation
Organization	Organization

The Project was evaluated by us on DD/MM/YYYY

EXAMINER 1	EXAMINER 2
EXAMINER 3	EXAMINER 4

ACKNOWLEDGEMENTS

We students of Computer Science and System Engineering currently in our 4th-year in KIIT UNIVERSITY, are preparing a major project named “HealthCare ChatBot”. We express our sincere gratitude to Prof. JYOTIPRAKASH MISHRA who guided us for the completion of the project. We are grateful to all of our teachers for teaching us important aspects of this project's topics. We are enormously thankful to Jyotiprakash Sir, as without his inspiration and valuable guidance it would not have been possible to accomplish the project within the prescribed time. We would like to express our deep gratitude for generous support by our friends and online resources from Github and Youtube.

Priyansh Choudhary
Shweta Nathany
Debalina Mazumder

ABSTRACT

Healthcare is more crucial to begin a good life. However, obtaining a doctor's consultation in the event of a health problem is quite difficult. The proposed concept is to use Artificial Intelligence to construct a health care chatbot system that can diagnose diseases and deliver basic information about them before contacting a doctor. The technology offers text support and allows you to communicate with the bot in a user-friendly manner. The bot will tell you what type of sickness you have based on your symptoms and the doctor data that emerge in relation to your disease analgesics. It will also give you diet suggestions, which indicates what kind of food you should eat. The chatbot will ask a series of questions to clarify the user's symptoms, and the symptom confirmation will be completed. The sickness will be divided into two categories: minor and major. If it is a significant disease, the chatbot will provide the user with the contact information for a doctor as well as analgesics for further treatment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Health care has become a critical component of our daily lives. People nowadays are preoccupied with domestic and office tasks, and they are becoming increasingly addicted to the Internet. They don't pay much attention to their well-being. As a result, people avoid going to the doctor for minor issues. It has the potential to turn into a significant issue.

The goal of artificial intelligence (AI) is to replicate human cognitive functions. It's causing a paradigm shift in healthcare, thanks to the growing availability of healthcare data and the rapid advancement of analytics tools [1].

So, our concept is to develop an AI-powered health care chatbot system that can diagnose diseases and provide basic information about them before contacting a doctor. This allows people to learn more about their ailment and enhance their overall health. The user can obtain all kinds of disease information.

To respond to user enquiries, the system application employs a question and response protocol in the form of a chatbot. The answer to the question will depend on the user's request. The important keywords are extracted from the sentence and used to answer those sentences. If a match is found, a substantial response will be provided, or comparable responses will be displayed.

Based on user symptoms, the bot will diagnose the type of sickness you have and provide doctor details, analgesics, and nutritional recommendations. Using this application system, they may be able to lessen their health problems.

The method was created to help consumers save money and time on healthcare because they are unable to visit doctors or specialists when they are in need.

CHAPTER 2
BACKGROUND

2.1 Existing system:
Many existing systems feature live conversations via SMS, but they have significant limitations, such as no immediate response for patients who must wait a long time for specialists to acknowledge their request. Some processes may levy a fee for using live chat or telephonic communication.

There have even been some smartphone apps developed that can forecast the probability of a disease and offer a diagnosis to a certain person depending on their health circumstances [2]. Effective early stage diagnosis, on the other hand, is still regarded as a difficult task [3]. Many academics have recently begun to use deep-learning models to achieve much better results than machine learning models.

2.2 Proposed system:
In our proposed system, the user can communicate with the bot via text about their query.
To answer the queries, the system employs an expert system.
The user can also see which doctors are accessible for that ailment.
Multiple people can utilise this technology to receive online counselling sessions.
The Chat Bot will prescribe analgesics and nutritional recommendations based on the condition.

2.3 SYSTEM REQUIREMENTS & PREREQUISITES:

2.3.1 Hardware Requirements:

Hard Disk : 10GB (minimum free space)
RAM : 512MB (minimum)
Processor: Intel® CoreTM i3-2350M CPU @ 2.30GHz
System Type: 64-bit Operating System

2.3.2 Software Requirements:

2.3.2.1 Python 3: We have used Python which is statistical mathematical programming.
1. Python code is more compact and readable than MATLAB
2. The python data structure is superior to MATLAB
3. It is an open source and also provides more graphic packages and data sets.

Python Libraries Used:

- NLP (Natural Language Processing) / NLTK
- scikit-learn
- Numpy
- Gensim
- Pandas
- Tkinter
- Pillow

2.3.2.2 IDE: Pycharm 2012.2.3:

PyCharm is a similar tool that organizes code and helps to run the data science process. It even has Jupyter Notebook support — however, it is just available in the paid, professional version.

The benefits of PyCharm over IDE like Jupyter:

- Python development
- Git integration
- Code formatting
- PEP-9 styling
- Debugger feature
- Running scripts
- Unit testing
- More developers are probably used to working with this tool, so in terms of collation, this tool is more accessible between departments or roles of engineering

2.3.3 Algorithms Used for training:

Decision Trees:

Decision Trees follow Divide-and-Conquer Algorithm.

Since we had to predict a particular disease and not a continuous value we have used a decision tree classifier and not a regressor. Here the decision variable is Categorical/ discrete.

A basic model for classifying cases is Decision Tree Classifiers. It's supervised machine learning in which data is constantly separated according to a parameter.

We start at the root of the tree and split the data on the characteristic that yields the most information gain using the decision method (IG).

We can then continue this splitting operation at each child node in an iterative process until the leaves are pure. This indicates that all of the samples at each leaf node are of the same class.

To avoid overfitting, we may place a restriction on the depth of the tree in practice.

SVM (Support Vector Machine):

(SVM) is a supervised machine learning technique that may be applied to classification and regression tasks. It is, however, mostly employed to solve categorization difficulties. Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a certain coordinate in the SVM algorithm.

Then we accomplish classification by locating the hyper-plane that clearly distinguishes the two classes.

Simply put, support vectors are the coordinates of each individual observation.

The SVM classifier is a frontier that separates the two classes (hyper-plane/line) the most effectively.

2.3.4 Dataset Used for training:

Data used for Training model:

We have gathered the symptoms and prognosis data for the project using the Google API dataset. The Google Cloud Public Dataset provides more than 100 public datasets from different industries, allowing us to join this data with our own projects to precipitate new findings.

Data used for Symptoms Analysis:

We have used Google API's data for analysis of prognosis such as disease precautions to be taken, disease caused due to other symptoms and disease severity. This data helped us gain confidence level after predictions.

Data used for Hospitals and Medical Suggestions:

We have used Indian Government Hospitals data which has information about which hospital is located in which area assigned via pin codes and what are the fields of treatments provided there. This helps users to choose medical facilities in a hassle free manner.

CHAPTER 3
PROJECT ANALYSIS/ PROJECT IMPLEMENTATION

3.1 Requirement Analysis

Property	Measures
Speed	The Chatbot stays live until the user gives input. Model Training and prediction speeds are depending on the datasets. Very minimal time in this case.
Size	The code takes up around 8MB of space i.e 2^{23} Bytes of space.
Ease of Use	Users can use this platform at any time independent of the network. The only requirement is python Installed to run the executable.
Reliability	The project stands at 82% accuracy from the pre-imported dataset. Also the model gives you output of how much percentage of surety is in the prediction of disease/ prognosis.
Robustness	The model handles several cases where exceptions may arise while the user inputs his/her responses and also is independent of any network requirements as it exists locally. Only chances where it fails are when hardware failure occurs.

Table 3.1 Requirement Analysis

3.2 System Design

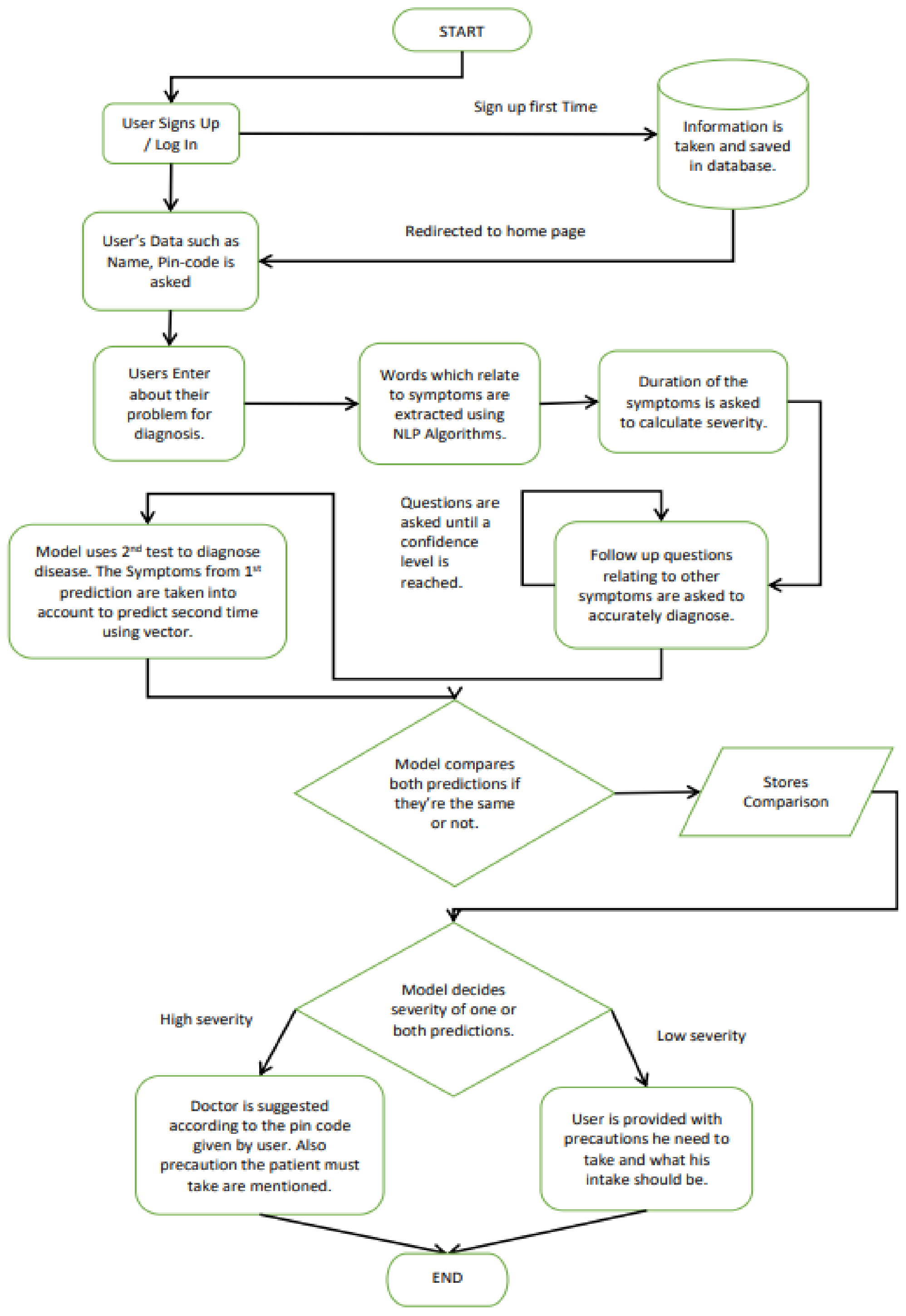

START
User Signs Up / Log In
Sign up first Time
Information is taken and saved in database.
Redirected to home page
User's Data such as Name, Pin-code is asked
Users Enter about their problem for diagnosis.
Words which relate to symptoms are extracted using NLP Algorithms.
Duration of the symptoms is asked to calculate severity.
Questions are asked until a confidence level is reached.
Follow up questions relating to other symptoms are asked to accurately diagnose.
Model uses 2nd test to diagnose disease. The Symptoms from 1st prediction are taken into account to predict second time using vector.
Model compares both predictions if they're the same or not.
Stores Comparison
Model decides severity of one or both predictions.
High severity
Low severity
Doctor is suggested according to the pin code given by user. Also precaution the patient must take are mentioned.
User is provided with precautions he need to take and what his intake should be.
END

Figure 3.2.1 Workflow System Design

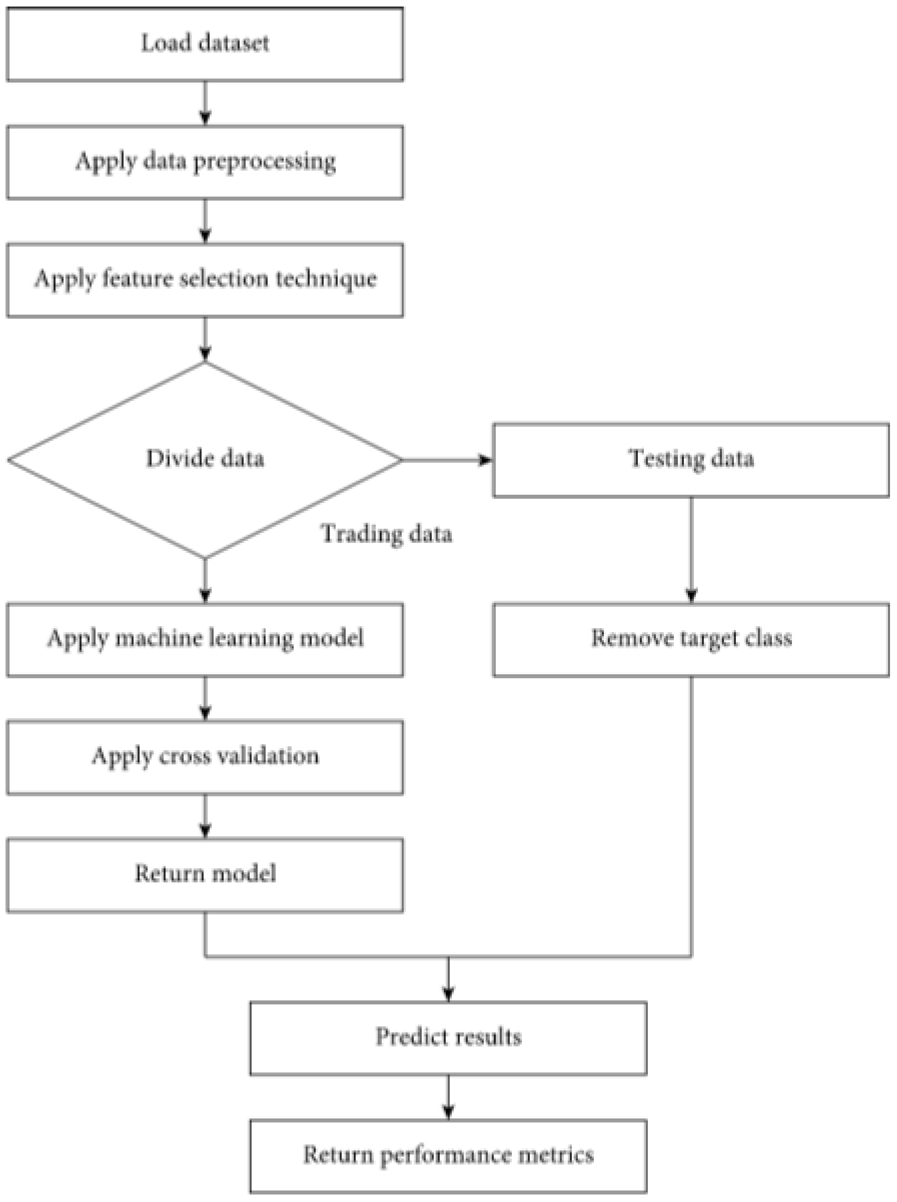


Figure 3.2.2 Prediction System Design

3.3 System Testing

The model takes into account the various symptoms and tries to predict the disease/prognosis . The model also gives confidence level as to confirm what percentage of its prediction might be true. Later prognosis and its precautions, treatment is shown to them.

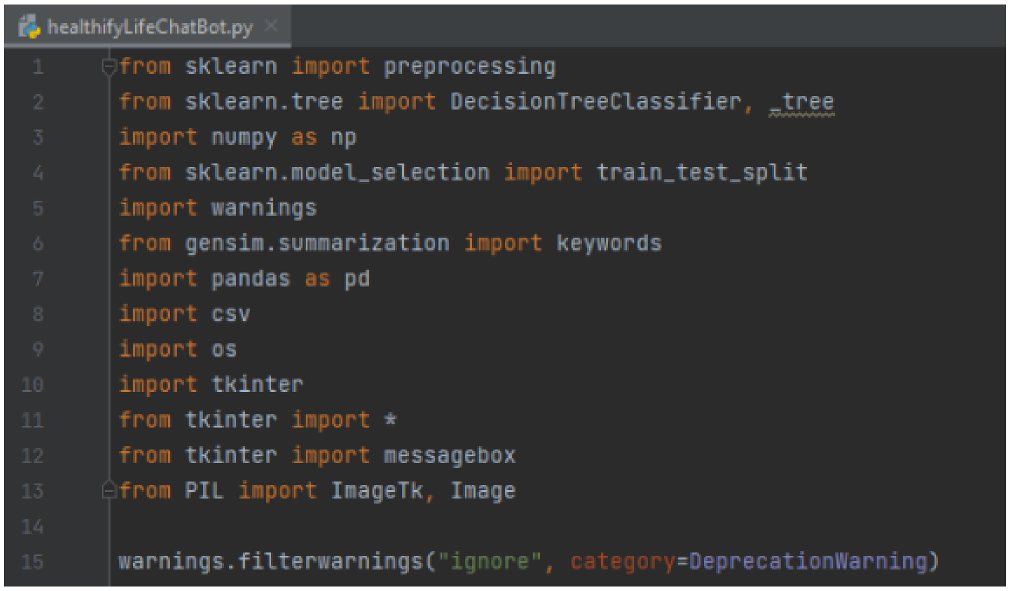
Test ID	Test Case Title	Test Condition	System Behaviour	Expected Result
T01	Prediction 01	chills, High Fever, Vomiting, Headache, Nausea	Malaria (Confidence Level 0.62)	Malaria
T02	Prediction 02	constipation, pain during bowel movements,pain in anal region, irritation in anus	Dimorphic hemorrhoids (piles) (Confidence Level 0.8)	Piles
T03	Prediction 03	Joint pain, muscle weakness, movement stiffness, swelling joints	Osteoarthritis or Arthritis (Confidence Level 0.4)	Arthritis

Table 3.3 System Testing

3.4 Implementation

Step 1 :

Importing all the required libraries.



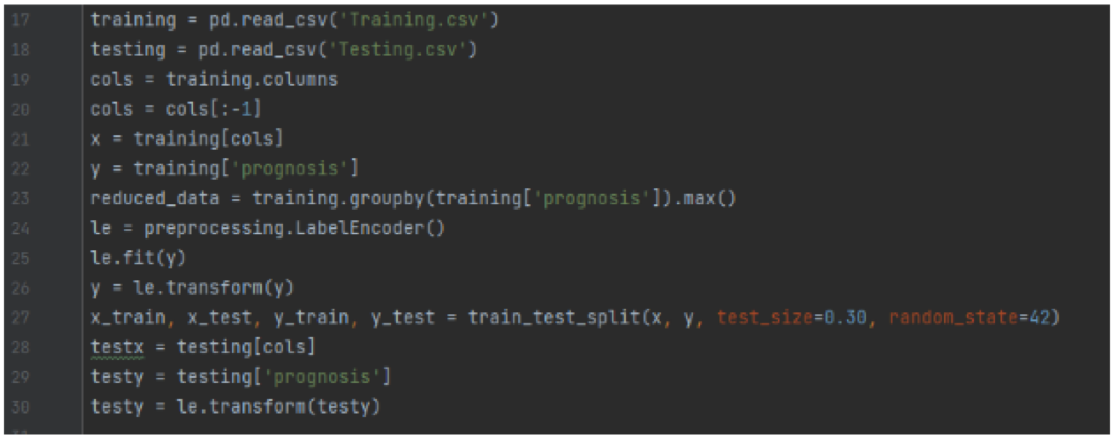
```
healthifyLifeChatBot.py
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier, _tree
import numpy as np
from sklearn.model_selection import train_test_split
import warnings
from gensim.summarization import keywords
import pandas as pd
import csv
import os
import tkinter
from tkinter import *
from tkinter import messagebox
from PIL import ImageTk, Image

warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Figure 3.4.1

Step 2 :

The training and testing csv files are read. All the symptoms columns from the training file are stored in *x* while the *prognosis* column is stored in *y*. Since, our machine learning model can not process String columns, we use label encoding and fit transform the *prognosis* column in the train data and transform it accordingly in the test data as well.

After basic preprocessing, such as removing redundant data and anomalies, the whole training data set is again split into *x_train, x_test, y_test* and *y_train*, for training our model and to check the accuracy of its prediction.

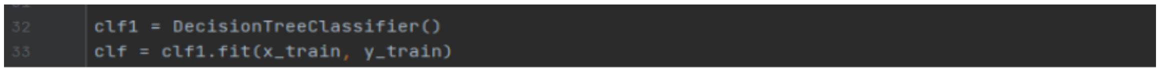


```
training = pd.read_csv('Training.csv')
testing = pd.read_csv('Testing.csv')
cols = training.columns
cols = cols[:-1]
x = training[cols]
y = training['prognosis']
reduced_data = training.groupby(training['prognosis']).max()
le = preprocessing.LabelEncoder()
le.fit(y)
y = le.transform(y)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=42)
testx = testing[cols]
testy = testing['prognosis']
testy = le.transform(testy)
```

Figure 3.4.2

Step 3 :

Next, we fit the decision tree on our *x_train* and *y_train* data.



```
clf1 = DecisionTreeClassifier()
clf = clf1.fit(x_train, y_train)
```

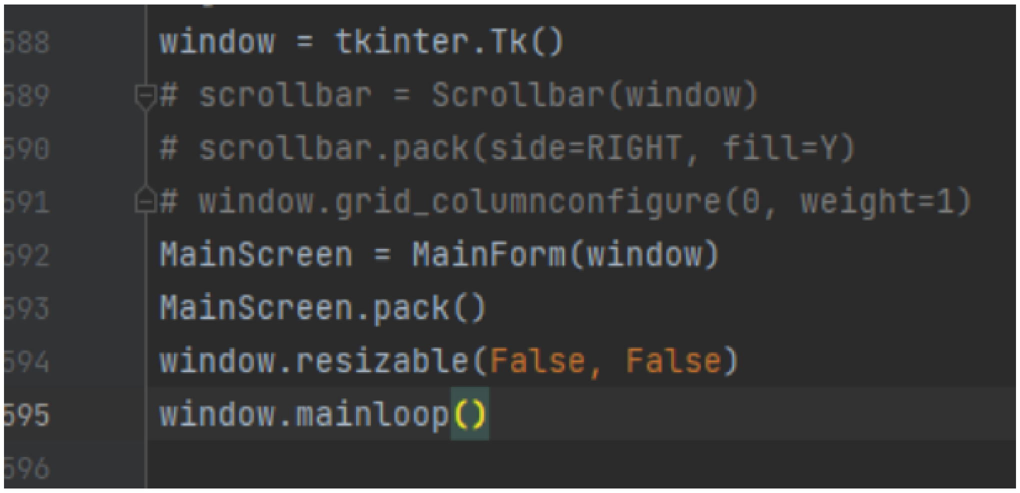
Figure 3.4.3

Step 4 :

Multiple dictionaries are created which extract the details from the csv files used as mentioned in 2.3.4.

Step 5 :

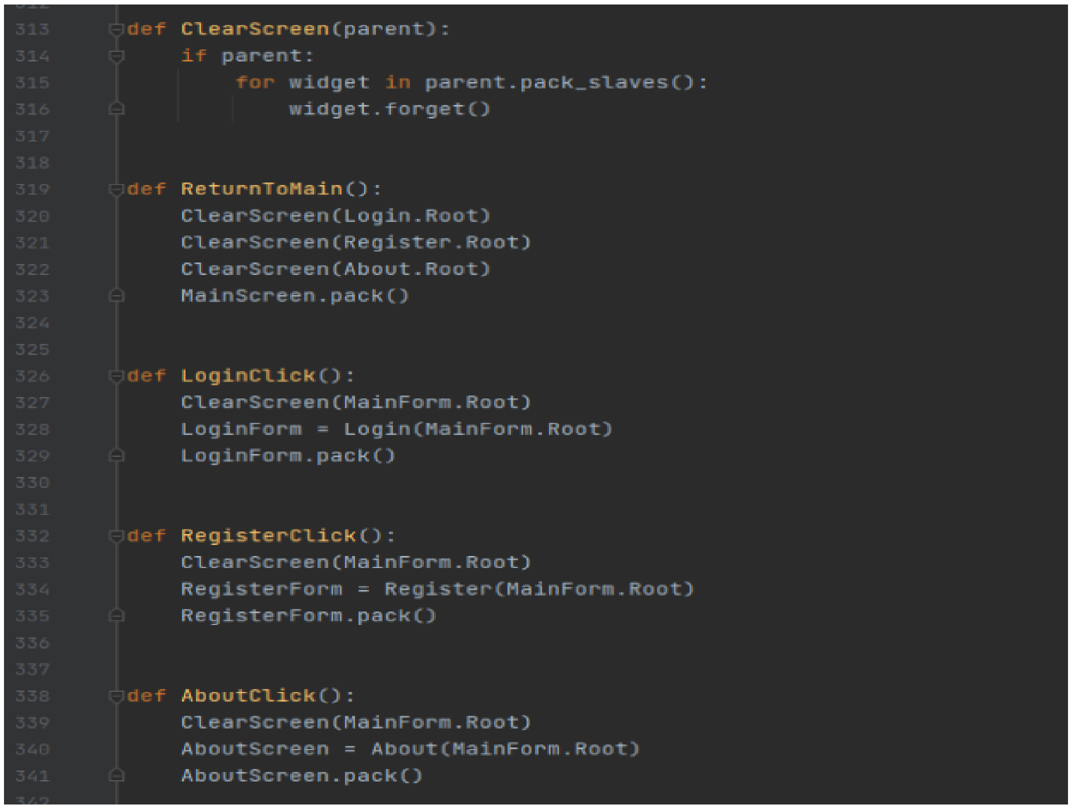
Upon execution, the main tkinter window pops up.



```
window = tkinter.Tk()
# scrollbar = Scrollbar(window)
# scrollbar.pack(side=RIGHT, fill=Y)
# window.grid_columnconfigure(0, weight=1)
MainScreen = MainForm(window)
MainScreen.pack()
window.resizable(False, False)
window.mainloop()
```

Figure 3.4.4

Here, a user can login, register, read about the chat-bot or can simply exit from the application.



```
def ClearScreen(parent):
    if parent:
        for widget in parent.pack_slaves():
            widget.forget()


def ReturnToMain():
    ClearScreen(Login.Root)
    ClearScreen(Register.Root)
    ClearScreen(About.Root)
    MainScreen.pack()


def LoginClick():
    ClearScreen(MainForm.Root)
    LoginForm = Login(MainForm.Root)
    LoginForm.pack()


def RegisterClick():
    ClearScreen(MainForm.Root)
    RegisterForm = Register(MainForm.Root)
    RegisterForm.pack()


def AboutClick():
    ClearScreen(MainForm.Root)
    AboutScreen = About(MainForm.Root)
    AboutScreen.pack()
```

Figure 3.4.5

An existing user can log in with their credentials or a new user can register themselves by entering a username and password which is then stored locally.

Step 6 :

After signing in, the User is then redirected to the diagnosis page where the user can start their diagnosis.

Step 7 :

The user is then prompted to describe their symptoms from which the keywords are extracted using NLP, TF algorithms.

We are using the gensim library for keyword extraction. Tokenization is performed on the entered text, followed by stemming and lemmatization.

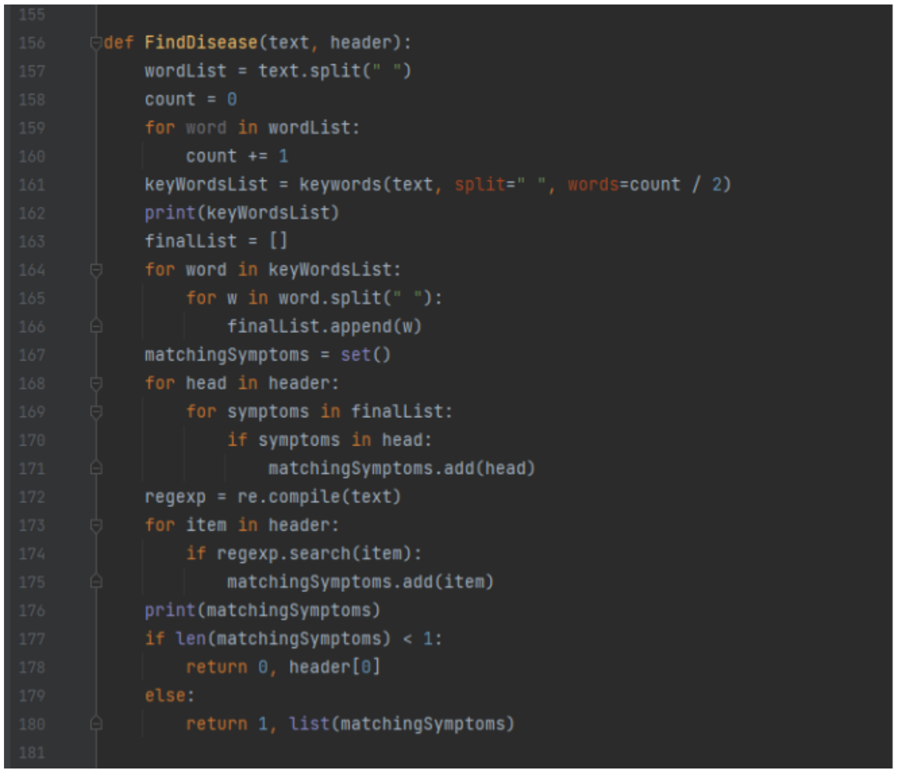
The TF*IDF algorithm is used to weigh a keyword in any content and assign importance to that keyword based on the number of times it appears in the document.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document). IDF: Inverse Document Frequency, which measures how important a term is.

$$\begin{aligned}\mathrm{idf} &= -\log P(t|D)\\ &= \log \frac{1}{P(t|D)}\\ &= \log \frac{N}{|\{d \in D : t \in d\}|}\end{aligned}$$

Figure 3.4.6

The extracted keywords are then compared with the column heads, upon matching the symptoms are appended to the *matchingSymptoms*



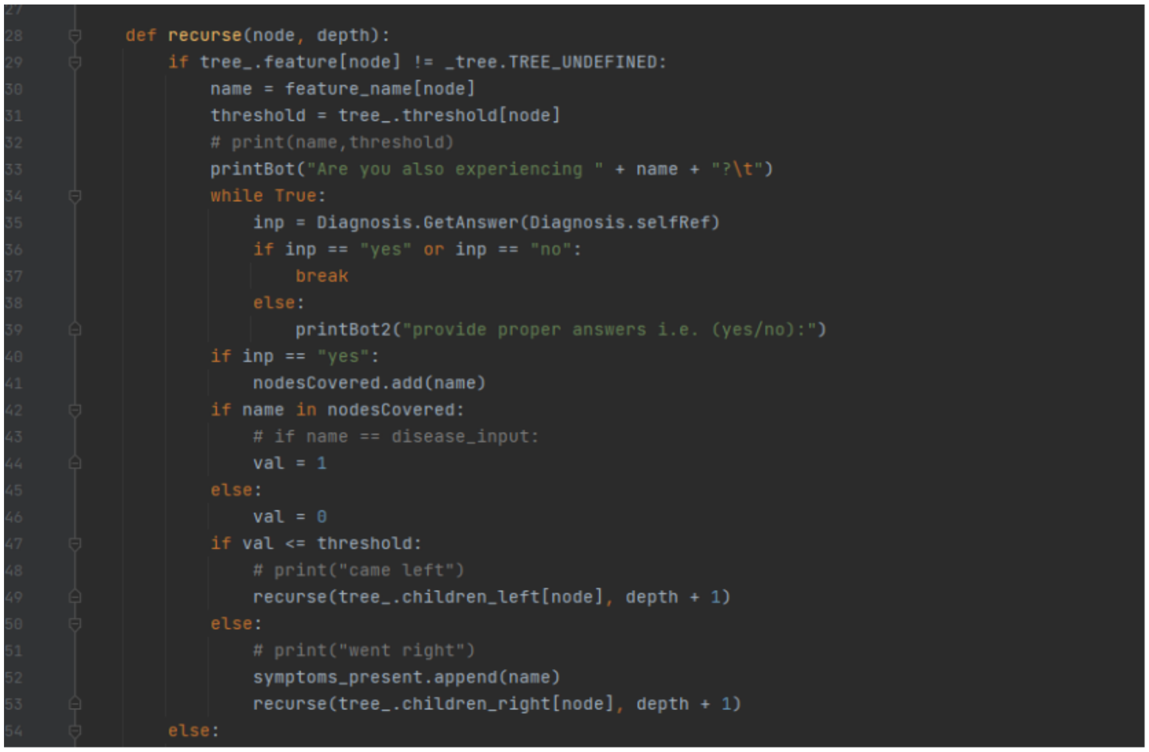
```python
def FindDisease(text, header):
    wordList = text.split(" ")
    count = 0
    for word in wordList:
        count += 1
    keyWordsList = keywords(text, split=" ", words=count / 2)
    print(keyWordsList)
    finalList = []
    for word in keyWordsList:
        for w in word.split(" "):
            finalList.append(w)
    matchingSymptoms = set()
    for head in header:
        for symptoms in finalList:
            if symptoms in head:
                matchingSymptoms.add(head)
    regexp = re.compile(text)
    for item in header:
        if regexp.search(item):
            matchingSymptoms.add(item)
    print(matchingSymptoms)
    if len(matchingSymptoms) < 1:
        return 0, header[0]
    else:
        return 1, list(matchingSymptoms)
```

Figure 3.4.7

Step 8 :

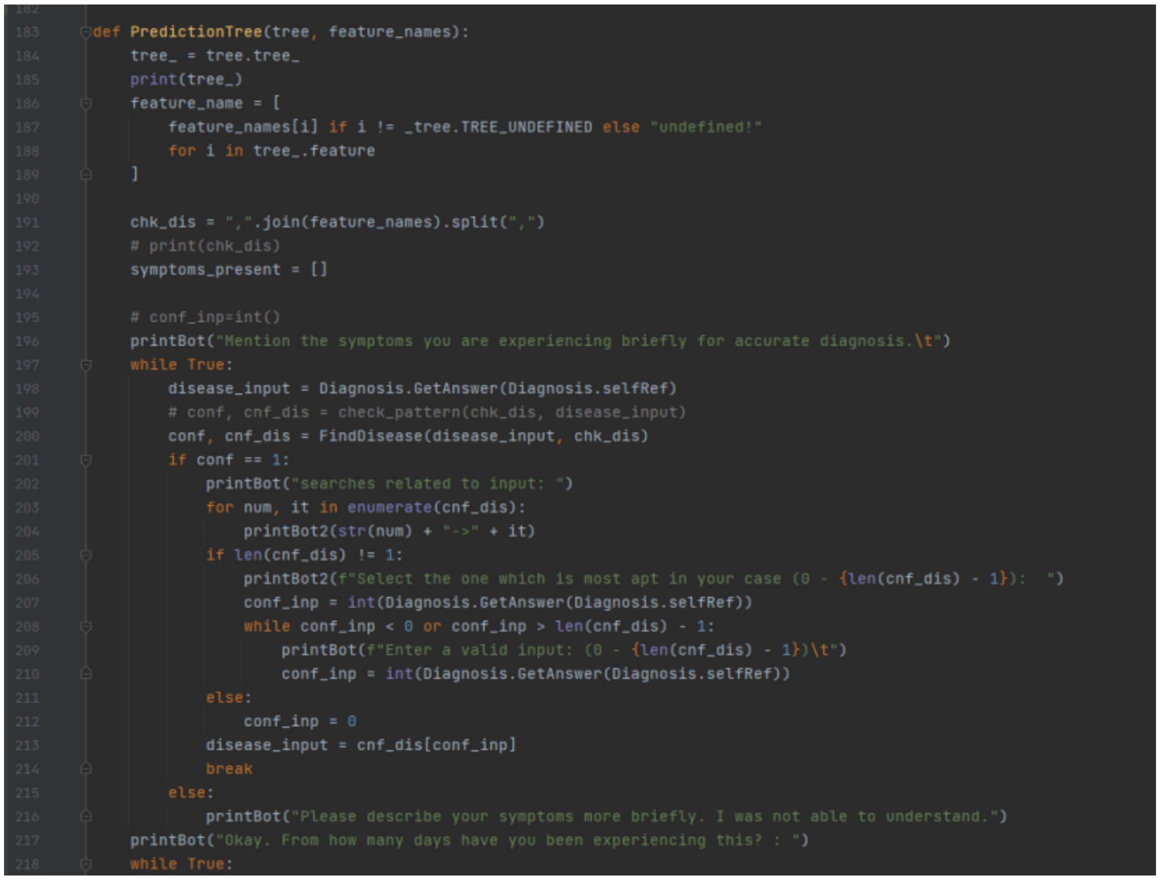
We are using _tree from scikit learn to extract internal logic of the decision tree classifier. We start from the root node, where every node other than the leaf node is a decision node.

The user is then asked a series of yes/no questions to traverse the tree until we reach a leaf node which is our predicted *prognosis*.



```
    def recurse(node, depth):
        if tree_.feature[node] != _tree.TREE_UNDEFINED:
            name = feature_name[node]
            threshold = tree_.threshold[node]
            # print(name,threshold)
            printBot("Are you also experiencing " + name + "?\t")
            while True:
                inp = Diagnosis.GetAnswer(Diagnosis.selfRef)
                if inp == "yes" or inp == "no":
                    break
                else:
                    printBot2("provide proper answers i.e. (yes/no):")
            if inp == "yes":
                nodesCovered.add(name)
            if name in nodesCovered:
                # if name == disease_input:
                val = 1
            else:
                val = 0
            if val <= threshold:
                # print("came left")
                recurse(tree_.children_left[node], depth + 1)
            else:
                # print("went right")
                symptoms_present.append(name)
                recurse(tree_.children_right[node], depth + 1)
        else:
```

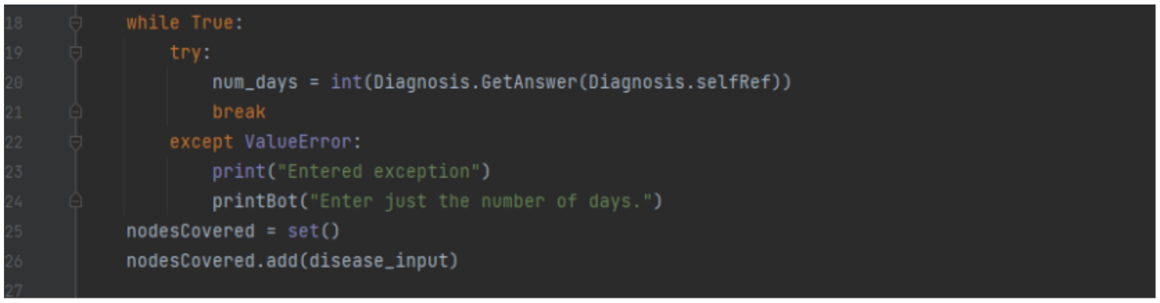
Figure 3.4.8



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

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

    # conf_inp=int()
    printBot("Mention the symptoms you are experiencing briefly for accurate diagnosis.\t")
    while True:
        disease_input = Diagnosis.GetAnswer(Diagnosis.selfRef)
        # conf, cnf_dis = check_pattern(chk_dis, disease_input)
        conf, cnf_dis = FindDisease(disease_input, chk_dis)
        if conf == 1:
            printBot("searches related to input: ")
            for num, it in enumerate(cnf_dis):
                printBot2(str(num) + "->" + it)
            if len(cnf_dis) != 1:
                printBot2(f"Select the one which is most apt in your case (0 - {len(cnf_dis) - 1}):  ")
                conf_inp = int(Diagnosis.GetAnswer(Diagnosis.selfRef))
                while conf_inp < 0 or conf_inp > len(cnf_dis) - 1:
                    printBot(f"Enter a valid input: (0 - {len(cnf_dis) - 1})\t")
                    conf_inp = int(Diagnosis.GetAnswer(Diagnosis.selfRef))
            else:
                conf_inp = 0
            disease_input = cnf_dis[conf_inp]
            break
        else:
            printBot("Please describe your symptoms more briefly. I was not able to understand.")
    printBot("Okay. From how many days have you been experiencing this? : ")
    while True:
```

Figure 3.4.9

Follow up questions are asked such as the duration of certain symptoms, whose values are stored to later determine the severity of the disease.

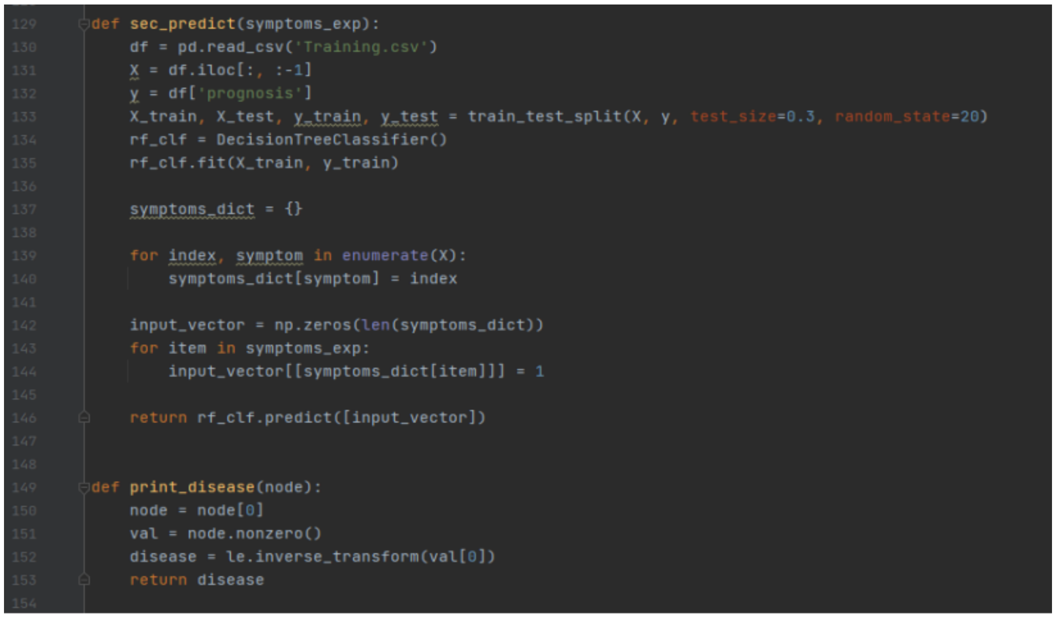


```python
    while True:
        try:
            num_days = int(Diagnosis.GetAnswer(Diagnosis.selfRef))
            break
        except ValueError:
            print("Entered exception")
            printBot("Enter just the number of days.")
    nodesCovered = set()
    nodesCovered.add(disease_input)
```

Figure 3.4.10

Step 9 :

After a certain confidence level is achieved, our model uses a second test to again diagnose the disease. Here, the symptoms from the first prediction are taken into consideration to re-predict the disease.

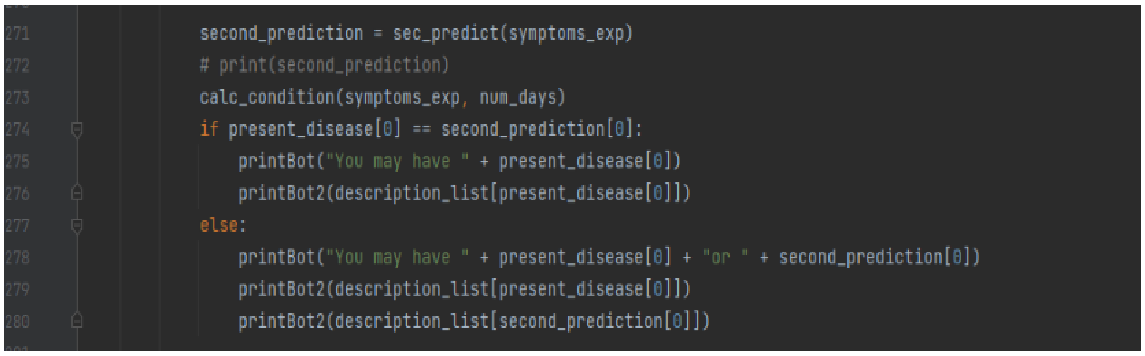


```python
def sec_predict(symptoms_exp):
    df = pd.read_csv('Training.csv')
    X = df.iloc[:, :-1]
    y = df['prognosis']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=20)
    rf_clf = DecisionTreeClassifier()
    rf_clf.fit(X_train, y_train)

    symptoms_dict = {}

    for index, symptom in enumerate(X):
        symptoms_dict[symptom] = index

    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 disease
```

Figure 3.4.11

Step 10 :

Both the predictions are now compared to check if the results are same or not.

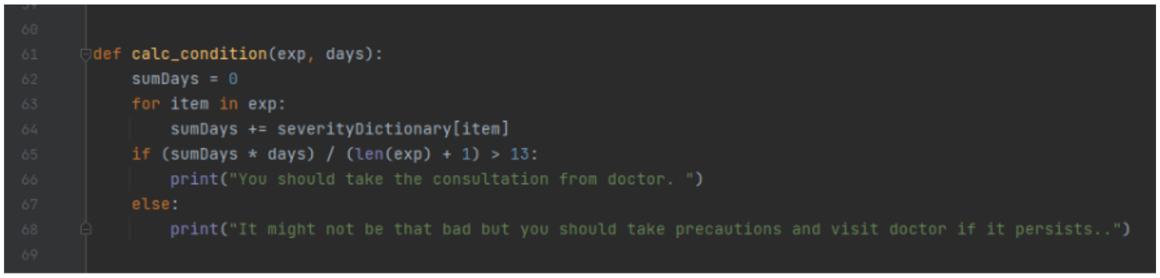


```python
            second_prediction = sec_predict(symptoms_exp)
            # print(second_prediction)
            calc_condition(symptoms_exp, num_days)
            if present_disease[0] == second_prediction[0]:
                printBot("You may have " + present_disease[0])
                printBot2(description_list[present_disease[0]])
            else:
                printBot("You may have " + present_disease[0] + "or " + second_prediction[0])
                printBot2(description_list[present_disease[0]])
                printBot2(description_list[second_prediction[0]])
```

Figure 3.4.12

Step 11 :

According to the severity level the user is suggested precautionary measures :



```python
def calc_condition(exp, days):
    sumDays = 0
    for item in exp:
        sumDays += severityDictionary[item]
    if (sumDays * 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 and visit doctor if it persists..")
```

Figure 3.4.13

Here, *sumDays* stores the sum of the value associated with each symptom in the “Symptom_severity” csv file.

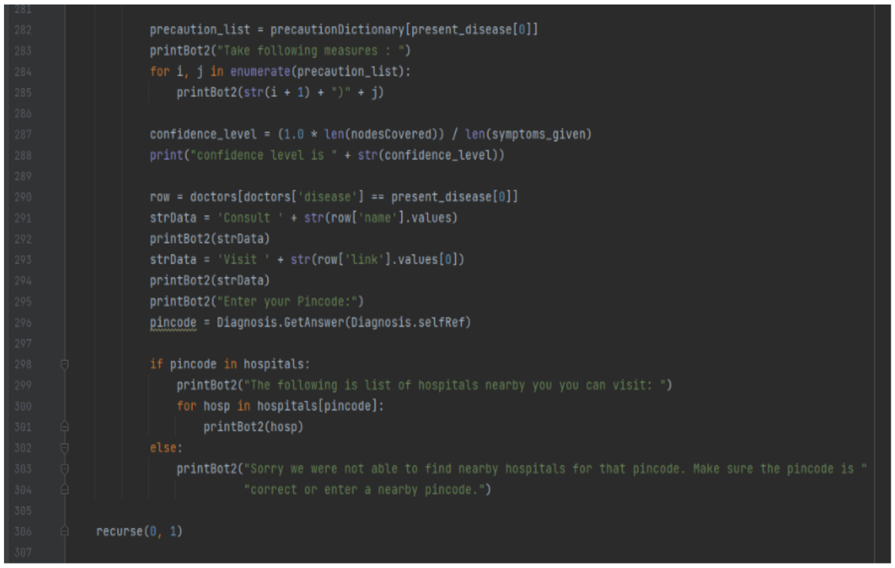
If the value returned by these formulas is greater than 13, then it is categorized as a high severity case.

High severity :

Customary precautions are suggested along with the name and contact details of a doctor that is nearest to the patients’ location. This is again calculated by our model with the help of the pin code that was provided by the user at the time of log in.

Low severity :

Precautionary measures are suggested like over the counter medicines, food intake for speedy recovery, along with basic at-home remedies like taking steam, hold/cold compress etc. for quick relief.



```python
            precaution_list = precautionDictionary[present_disease[0]]
            printBot2("Take following measures : ")
            for i, j in enumerate(precaution_list):
                printBot2(str(i + 1) + ")" + j)

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

            row = doctors[doctors['disease'] == present_disease[0]]
            strData = 'Consult ' + str(row['name'].values)
            printBot2(strData)
            strData = 'Visit ' + str(row['link'].values[0])
            printBot2(strData)
            printBot2("Enter your Pincode:")
            pincode = Diagnosis.GetAnswer(Diagnosis.selfRef)

            if pincode in hospitals:
                printBot2("The following is list of hospitals nearby you you can visit: ")
                for hosp in hospitals[pincode]:
                    printBot2(hosp)
            else:
                printBot2("Sorry we were not able to find nearby hospitals for that pincode. Make sure the pincode is "
                          "correct or enter a nearby pincode.")

    recurse(0, 1)
```

Figure 3.4.14

CHAPTER 4
RESULTS & DISCUSSIONS

4.1 Screenshots of the Project

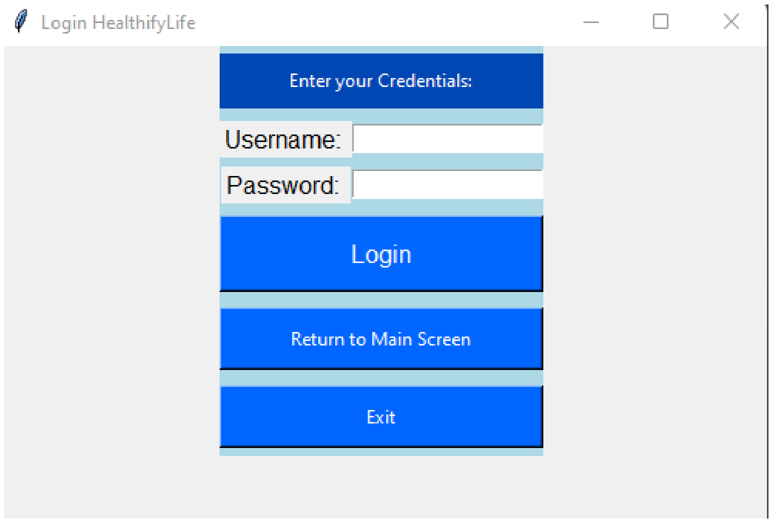


Fig 4.1 Login and Registration Page

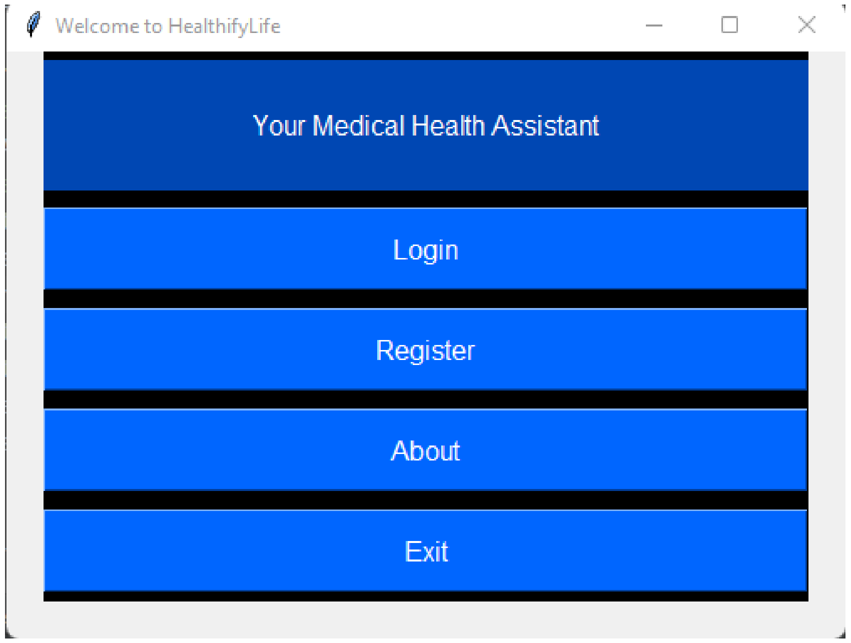
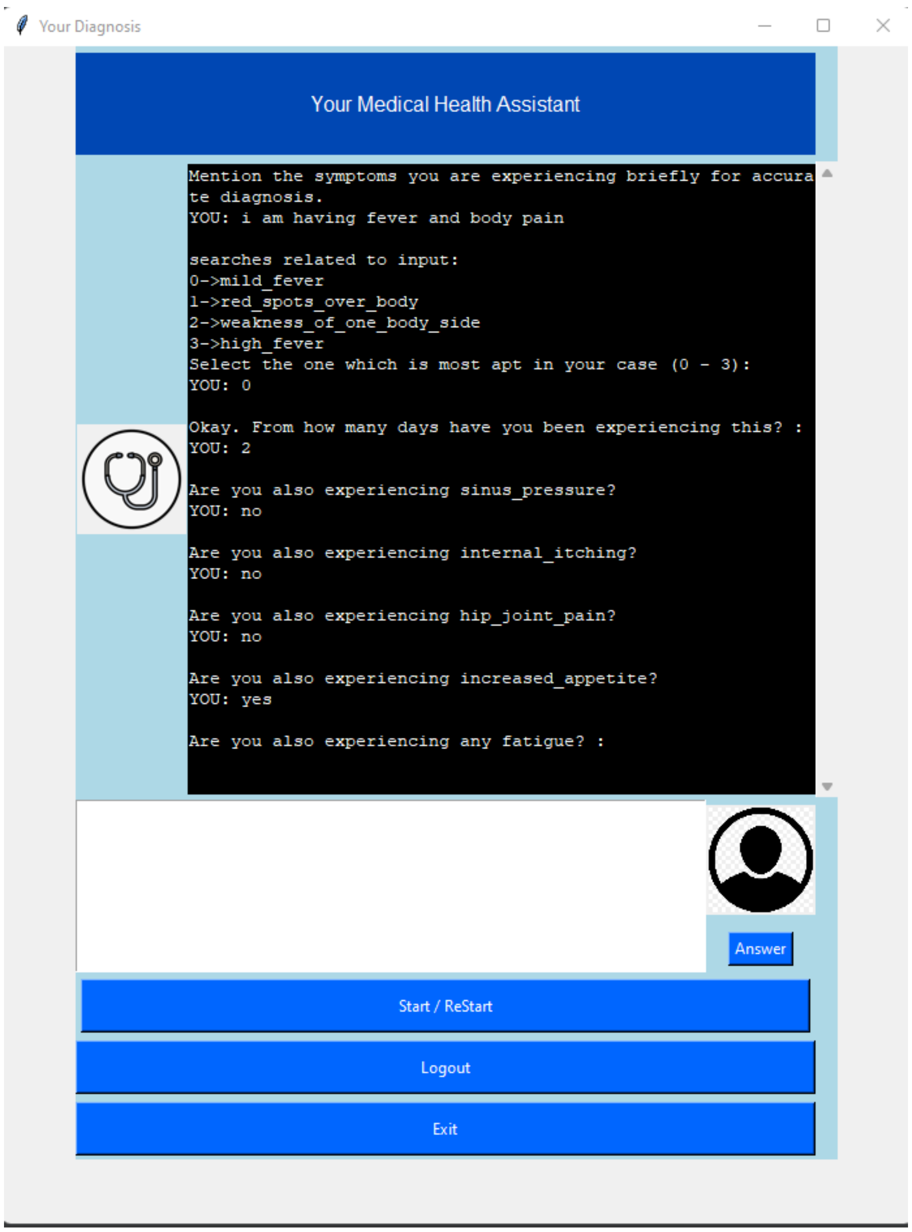
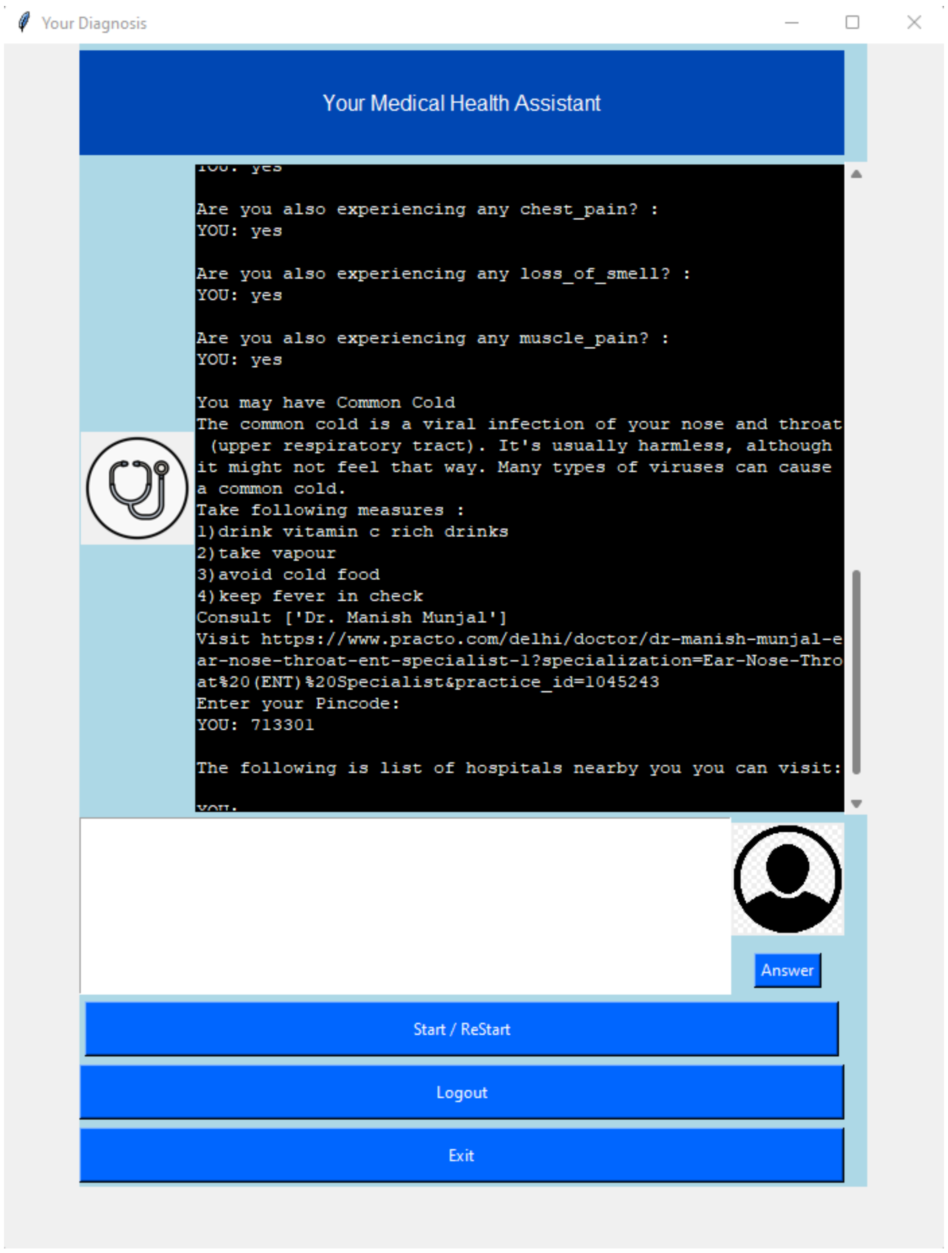


Fig 4.2 Main Menu



```
Mention the symptoms you are experiencing briefly for accura
te diagnosis.
YOU: i am having fever and body pain

searches related to input:
0->mild_fever
1->red_spots_over_body
2->weakness_of_one_body_side
3->high_fever
Select the one which is most apt in your case (0 - 3):
YOU: 0

Okay. From how many days have you been experiencing this? :
YOU: 2

Are you also experiencing sinus_pressure?
YOU: no

Are you also experiencing internal_itching?
YOU: no

Are you also experiencing hip_joint_pain?
YOU: no

Are you also experiencing increased_appetite?
YOU: yes

Are you also experiencing any fatigue? :
```

Fig 4.3 Conversation window between the user and the chatbot



```
YOU: yes

Are you also experiencing any chest_pain? :
YOU: yes

Are you also experiencing any loss_of_smell? :
YOU: yes

Are you also experiencing any muscle_pain? :
YOU: yes

You may have Common Cold
The common cold is a viral infection of your nose and throat
 (upper respiratory tract). It's usually harmless, although
it might not feel that way. Many types of viruses can cause
a common cold.
Take following measures :
1)drink vitamin c rich drinks
2)take vapour
3)avoid cold food
4)keep fever in check
Consult ['Dr. Manish Munjal']
Visit https://www.practo.com/delhi/doctor/dr-manish-munjal-e
ar-nose-throat-ent-specialist-1?specialization=Ear-Nose-Thro
at%20(ENT)%20Specialist&practice_id=1045243
Enter your Pincode:
YOU: 713301

The following is list of hospitals nearby you you can visit:

YOU:
```

Fig 4.4 Diagnosis and advice made by the chatbot

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

Healthcare chatbots are the future of medicine since they help to reduce the amount of physical contact between patients and doctors in today's ever-increasing population. Our chatbot assists healthcare providers in their work and helps them improve their performance by communicating with patients in a human-like manner.

Through this chatbot, patients can get medical information, diagnoses at the first indication of a disease and location of the healthcare facility nearest to them. Our chatbot interacts with the user via natural language processing. It recognises user input using pattern matching and responds appropriately from the dataset given.

In the near future, the proposed model could be applied to other applications such as handwritten recognition [4], image filtering [5], cancer classification [6], and medical image segmentation [7], and the initial parameters of the proposed machine learning models could be tuned using various meta-heuristic techniques [8].

5.2 Future Work

This project's future scope is broad and far-reaching. Understanding the difficulties that health care chatbots face may help improve their performance, and they can be used in a range of areas, including education, healthcare, and business. The chatbot could be upgraded to work as a voice or face recognizer and engage with the patient or user on a more personal level, allowing the user to communicate with the chatbot using any language interface other than text or buttons.

5.3 Planning and Project Management

The Gantt chart is shown below:

	Week 1 & 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Literature review	🟧						
Finalizing problem		🟧					
Materials and methods			🟧				
Re-implementing the model				🟧			
Experimental results					🟧		
Preparation of project report						🟧	🟧

Fig 5.1: Gantt Chart

REFERENCES

[1] F. Jiang, Y. Jiang, H. Zhi et al., “Artificial intelligence in healthcare: past, present and future,” Stroke and Vascular Neurology, vol. 2, no. 4, pp. 230–243, 2017.

[2] Sidey-Gibbons, A. M. Jenni, and C. J. Sidey-Gibbons, “Machine learning in medicine: a practical introduction,” BMC Medical Research Methodology, vol. 19, 2019.

[3] J. Keto, H. Ventola, J. Jokelainen et al., “Cardiovascular disease risk factors in relation to smoking behaviour and history: a population-based cohort study,” Open Heart, vol. 3, no. 2, 2016.

[4] S. Ghosh, P. Shivakumara, P. Roy, U. Pal, and T. Lu, “Graphology based handwritten character analysis for human behaviour identification,” CAAI Transactions on Intelligence Technology, vol. 5, no. 1, pp. 55–65, 2020.

[5] H. S. Basavegowda, G. Dagnew, and G. Dagnew, “Deep learning approach for microarray cancer data classification,” CAAI Transactions on Intelligence Technology, vol. 5, no. 1, pp. 22–33, 2020.

[6] B. Gupta, M. Tiwari, and S. Singh Lamba, “Visibility improvement and mass segmentation of mammogram images using quantile separated histogram equalisation with local contrast enhancement,” CAAI Transactions on Intelligence Technology, vol. 4, no. 2, pp. 73–79, 2019.

[7] S. Osterland and J. Weber, “Analytical analysis of single-stage pressure relief valves,” International Journal of Hydromechatronics, vol. 2, no. 1, pp. 32–53, 2019.

[8] R. Wang, H. Yu, G. Wang, G. Zhang, and W. Wang, “Study on the dynamic and static characteristics of gas static thrust bearing with micro-hole restrictors,” International Journal of Hydromechatronics, vol. 2, no. 3, pp. 189–202, 2019.

HEALTHCARE CHAT-BOT

PRIYANSH CHOUDHARY

1805589

Abstract: Healthcare is more crucial to begin a good life. However, obtaining a doctor's consultation in the event of a health problem is quite difficult. The proposed concept is to use Artificial Intelligence to construct a health care chatbot system that can diagnose diseases and deliver basic information about them before contacting a doctor. The technology offers text support and allows you to communicate with the bot in a user-friendly manner. The bot will tell you what type of sickness you have based on your symptoms and the doctor data that emerge in relation to your disease analgesics.

Individual contribution and findings: My main contribution towards this project included the logic development to predict the disease using decision tree classifier model's internal structure. Here, I traverse the nodes of the tree fitted with the model along with using a second prediction via a vector of symptoms given by user and calculate confidence levels of predicted prognosis. I have also worked with my teammate towards development of the UI of the Diagnosis screen of the chatbot using Tkinter Library of python where the user and bot interact with each other. I have coded up the functionality part of the buttons and widgets along with fixing bugs in other parts of the app.

Individual contribution to project report preparation: : For my contribution towards the report preparation, I have written and completed Chapter 1, Chapter 2 as well as the references. Along with it I also contributed to findings of system testing and analysis.

PChoudhary

Full Signature of Supervisor:

…………………………….

Full signature of the student:

Priyansh Choudhary

HEALTHCARE CHAT-BOT

SHWETA NATHANY

1806164

Abstract: Healthcare is more crucial to begin a good life. However, obtaining a doctor's consultation in the event of a health problem is quite difficult. The proposed concept is to use Artificial Intelligence to construct a health care chatbot system that can diagnose diseases and deliver basic information about them before contacting a doctor. The technology offers text support and allows you to communicate with the bot in a user-friendly manner. The bot will tell you what type of sickness you have based on your symptoms and the doctor data that emerge in relation to your disease analgesics.

Individual contribution and findings: I played a significant role in both the front and the back end of the project. For the front end, I primarily worked on developing the UI for the login page, the main screen and the registration page that pops up on launching the chat-bot using tkinter. And for the back end, I majorly worked on the extraction of the keywords from the text entered by the user, with the help of NLP and NLTK. It aided us in picking up the symptoms the users were suffering from.

Individual contribution to project report preparation: In the project report preparation I have written and completed Chapter 3.

Shweta Nathany

Full Signature of Supervisor:

Full signature of the student:

Shweta Nathany

HEALTHCARE CHAT-BOT

DEBALINA MAZUMDER

1828064

Abstract: Healthcare is more crucial to begin a good life. However, obtaining a doctor's consultation in the event of a health problem is quite difficult. The proposed concept is to use Artificial Intelligence to construct a health care chatbot system that can diagnose diseases and deliver basic information about them before contacting a doctor. The technology offers text support and allows you to communicate with the bot in a user-friendly manner. The bot will tell you what type of sickness you have based on your symptoms and the doctor data that emerge in relation to your disease analgesics.

Individual contribution and findings: I have contributed to both the front-end and the backend of this project. I have pre-processed the training dataset and removed outliers and anomalies. I have used label encoding for feature selection and then I have used different machine learning models such as Decision Tree and SVM(Support Vector Machine) to select the model which best fits the data and makes more accurate predictions. I have worked with my teammate to make the UI interface between the user and the chatbot. Also, I have made necessary changes in our code to fix bugs such as adding images to the diagnosis window.

Individual contribution to project report preparation: In the project report preparation I have written and completed Chapter 4 as well as Chapter 5.

Debalina Mazumder

Full Signature of Supervisor:

Full signature of the student:
Debalina Mazumder