A

MINOR PROJECT REPORT

On

Smart Doc: AI Powered Medical Consultant with Human Intellect to Support and Enhance People's Lives in Emergencies

Submitted in partial fulfillment of the requirement for the award of the degree of

B.TECH

in

COMPUTER SCIENCE AND ENGINEERING

Submitted By

NAME OF THE STUDENT : Reg. No. RA1811003040117

DEPT. OF COMPUTER SCIENCE & ENGINEERING

SRM Institute of Science & Technology

Vadapalani Campus,Chennai

November 2021

A

MINOR PROJECT REPORT

On

Smart Doc: AI Powered Medical Consultant with Human Intellect to Support and Enhance People's Lives in Emergencies

Submitted in partial fulfillment of the requirement for the award of the degree of

B.TECH

in

COMPUTER SCIENCE AND ENGINEERING

Submitted By

NAME OF THE STUDENT : Reg. No. RA1811003040117

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University *u/s 3 of UGC Act, 1956*

DEPT. OF COMPUTER SCIENCE & ENGINEERING

SRM Institute of Science & Technology

Vadapalani Campus,Chennai

November 2021

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report **"Smart Doc: AI Powered Medical Consultant with Human Intellect to Support and Enhance People's Lives in Emergencies"** is the bonafide work of "**M. Aishwarya"** who carried out the project work under my supervision.

SIGNATURE OF THE GUIDE	**SIGNATURE OF THE HOD**
Name of the Staff with Qualification	Dr.S.Prasanna Devi, B.E.,M.E., Ph.D., PGDHRM.,PDF(IISc)
Assistant Professor	Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
SRM Institute of Science & Technology	SRM Institute of Science & Technology
Vadapalani Campus	Vadapalani Campus

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, **C.Sabarinathan** for his valuable inputs, able guidance,encouragement,whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Dr .S.Prasanna Devi, for encouraging and allowing us to present the project on the topic "Smart Doc: AI Powered Medical Consultant with Human Intellect to Support and Enhance People's Lives in Emergencies " at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

We take this opportunity to thank all our faculty members, Dean, Dr.C.V.Jayakumar, and Management who have directly or indirectly helped our project. Last but not the least we express our thanks to our friends for their cooperation and support.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Tech-based self-service channels and digital health interventions have the potential to support the patients in their everyday life and health professionals likewise. The rise in artificial intelligence and innovation in digital technologies have paved the way for the medical systems to expand to meet the expectations of the people who need health care support especially during these kinds of unprecedented circumstances such as hay fever, flu, and other viral infections. Although there are scalable self-service channels such as Alexa, Google Assistant, Siri, yet they cannot be applied in medical care settings due to their lack of domain knowledge. Hence this project presents a healthcare chatbot called 'Smart Doc' created using Artificial Intelligence that can have a significant impact on the lives of people. This chatbot provides us with a human-system interaction with a user-friendly interface and it aims to solve the specific prerequisites of a person who needs healthcare suggestions before visiting the hospital. The chatbot system works on the inputs provided by the user (patient) and answers them accordingly. The functionalities of Smart Doc include primary healthcare services where a normal person with a mobile can be able to have interaction regarding the services. The chatbot enables the person to identify themselves by providing their age and contact number and then allows them to specify their symptoms. It also helps to schedule an appointment with the concerned doctor in a nearby hospital facility or to have an online discussion with the doctor regarding the treatment. This will reduce healthcare costs and improve accessibility to medical knowledge even for people living in rural areas while enhancing their lives at the same time.

CHAPTER- 1
INTRODUCTION

1.1 OVERVIEW

Artificial Intelligence is undoubtedly impacting the healthcare industry as the utilization of chatbots has become popular recently. Organizations are reaping benefits of these AI-enabled virtual agents for automating their routine procedures and provide clients the 24×7 attention in areas like payments, client service, and marketing. AI has come as a savior in the healthcare industry. From detecting diseases to using life-saving machines, AI is making strong new scopes across the industry. Common people are not medically trained for understanding the extremity of their diseases. This is where chatbots can be a great help. They gather prime data from patients and depending on the input, they give more data to patients regarding their conditions and recommend further steps also.

1.2 PROBLEM DEFINITION

To lead a better life healthcare is extremely much important. But it's very difficult to get the consultation with the doctor just in case of any health issues. The
user can achieve the real advantage of a chatbot only if it can diagnose all quite disease and supply necessary information. A text-to-text diagnosis bot engages patients in conversation about their medical issues and provides a personalized
diagnosis supported their symptoms. Hence, people will have a thought about their health and have the correct protection.

1.3 EXISTING SYSTEM

Safe drug Bot is a virtual assistant designed to help healthcare professionals in providing valuable guidance and monitoring. Babylon Health offers A.I. consultation based on personal medical history and common medical knowledge as well as live video consultation with a real doctor. Your .Md is a symptom checker free platform offers actionable health information based on highly accurate sources and lets the user make the best choices for his health. Ada Health can assess the user's health based on the indicated symptoms using its vast, A.I.-based database. Buoy Health chatbot thoroughly asks you about the details of your medical state and offers you various solutions and actionable steps to take.

1.4 PROPOSED SYSTEM

The Proposed System consists of a user friendly chat interface through which a user can communicate with the system. It focuses on a healthcare chatbot which analyses the user's symptoms through a conversation with the user. The Smart Doc can converse with the user via text or speech. The symptoms are then passed to a Machine Learning (ML) algorithm that has been trained to diagnose diseases based on symptoms. The chatbot can make a predictive diagnosis. This can assist in providing the initial response as well as guide the individual to a specialized healthcare professional. The project is developed for the user to save the user their time in consulting the doctors or experts for the healthcare solution. Here, the application is developed to provide quality of answers in a short period of time. The application is improved with the security and effectiveness upgrades by ensuring user protection and characters and retrieving answers consequently for the questions. It removes the burden from the answer provider by directly delivering the answer to the user using an expert system.

CHAPTER- 2
PROJECT WORK

2.1 REQUIREMENTS

The project was implemented using the following hardware and software requirements.

2.1.1 HARDWARE

- Processor- i3 Processor Based Computer or higher
- Memory: 1 GB
- Hard Disk: 50 GB

2.1.2 SOFTWARE

- Windows 7 or higher
- Google Chrome Browser
- Visual Studio Code
- My SQL

2.1.3 TOOLS AND TECHNOLOGIES

- Python
- Flask
- HTML5
- CSS5
- MySQL
- Javascript

2.2 MODULES

2.2.1 Data Gathering

Data preparation is the primary step for any machine learning problem. The dataset from Kaggle is used for the project. This dataset consists of a CSV file which will be used to train the model. There is a total of 133 columns in the dataset out of which 132 columns represent the symptoms and the last column is the prognosis.

2.2.2 Data Cleaning

Cleaning is the most important step in a machine learning project. The quality of the data determines the quality of a machine learning model. So it is always necessary to clean the data before feeding it to the model for training. In the dataset all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a label encoder.

2.2.3 Model Building

After gathering and cleaning the data, the data is ready and can be used to train a machine learning model. The cleaned data is used to train the Support Vector Classifier, Naive Bayes Classifier, and Random Forest Classifier. After training the three models, The diseases is predicted for the input symptoms by combining the predictions of all three models. This makes the overall prediction more robust and accurate.

2.2.4 Web Development

The Front end of the project is created using HTML and CSS. It includes a user-friendly chat interface in which the user can communicate with the bot via text or over voice. Its functioning is achieved using Javascript. When a user clicks or performs any action a javascript function is called depending on the id of the element on which the action was performed. It also uses jquery to get the input from the user and reply through the flask framework.

2.2.5 Integration

Flask is an important web framework in python which is used to create end-to-end projects. It is based on the jinja2 template engine which combines web templates along with a data source. The final aim is to render dynamic pages. This project is developed using the flask that integrates the python modules with the web page. It also connects to the MySQL database where the data collected from the user is stored.

2.3 SYSTEM ARCHITECTURE

The Figure (2.1) below represents the user interaction with the Smart Doc. The user's details will be collected and stored in the database for future references. The smart doc then identifies the symptoms from the user's input and analyzes it after which the system displays the disease based on the user symptoms and suggests the required treatment.

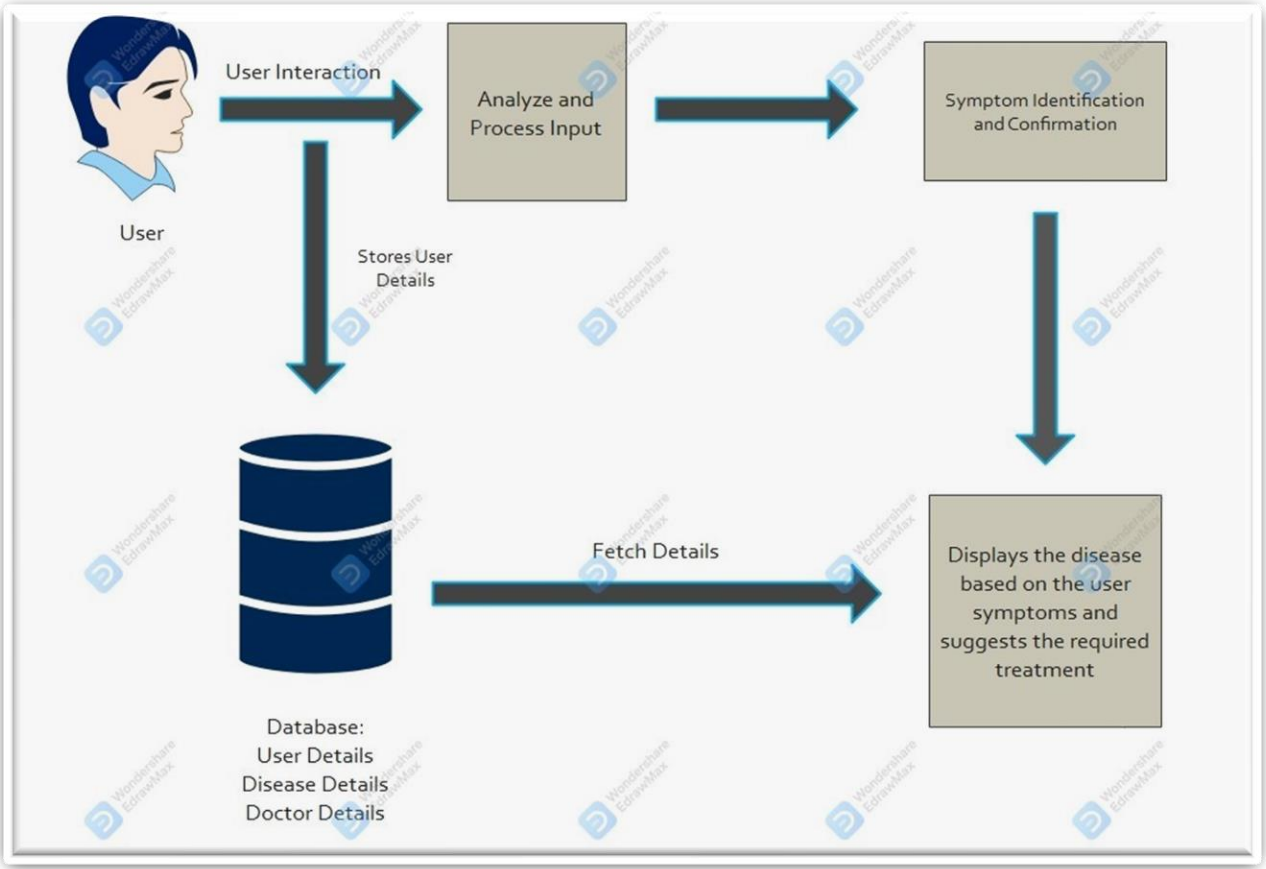

User
User Interaction
Analyze and Process Input
Symptom Identification and Confirmation
Stores User Details
Fetch Details
Displays the disease based on the user symptoms and suggests the required treatment
Database:
User Details
Disease Details
Doctor Details

Fig 2.1 Architechture Diagram

2.4 DIAGRAMS

Use Case Diagram:

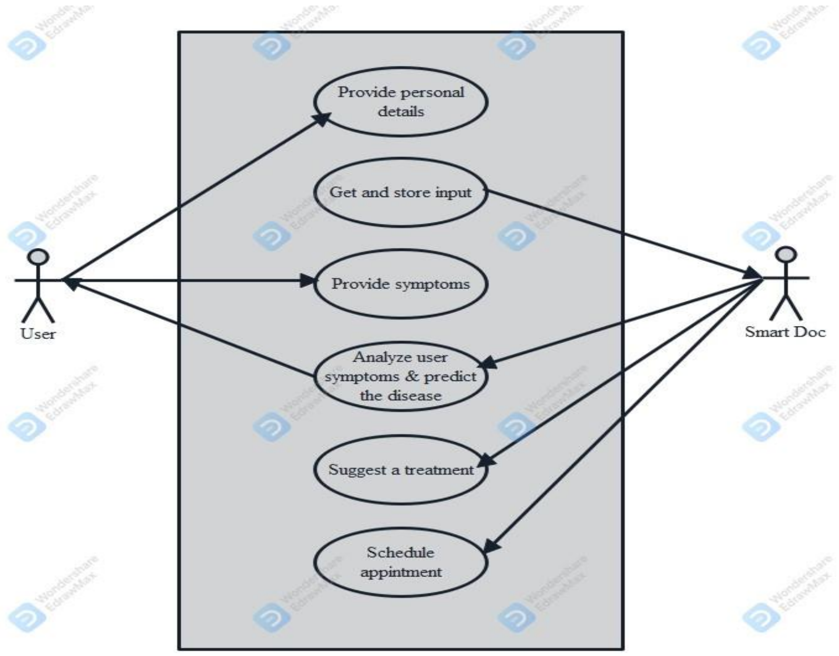


Fig 2.2 Use Diagram

Class Diagram:

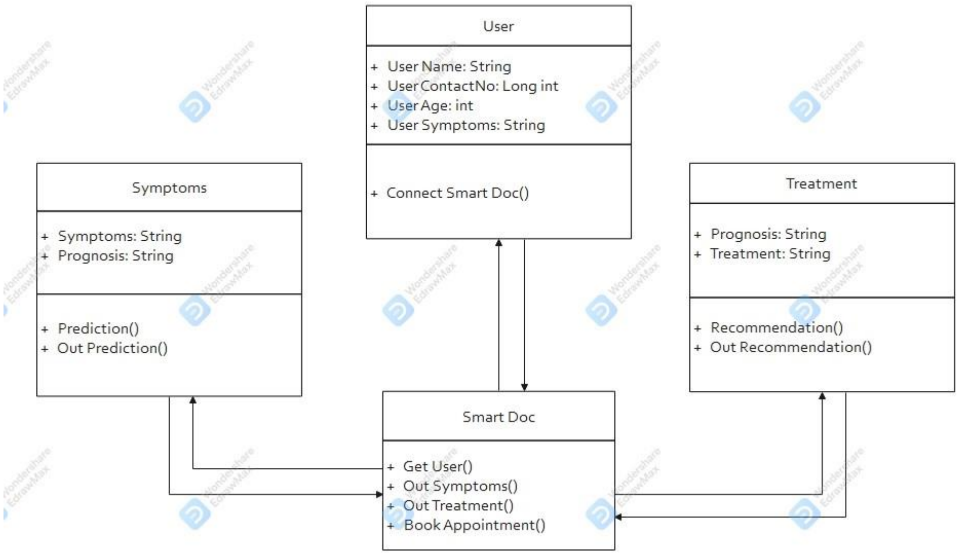


Fig 2.3 Class Diagram

Data Flow Diagram:

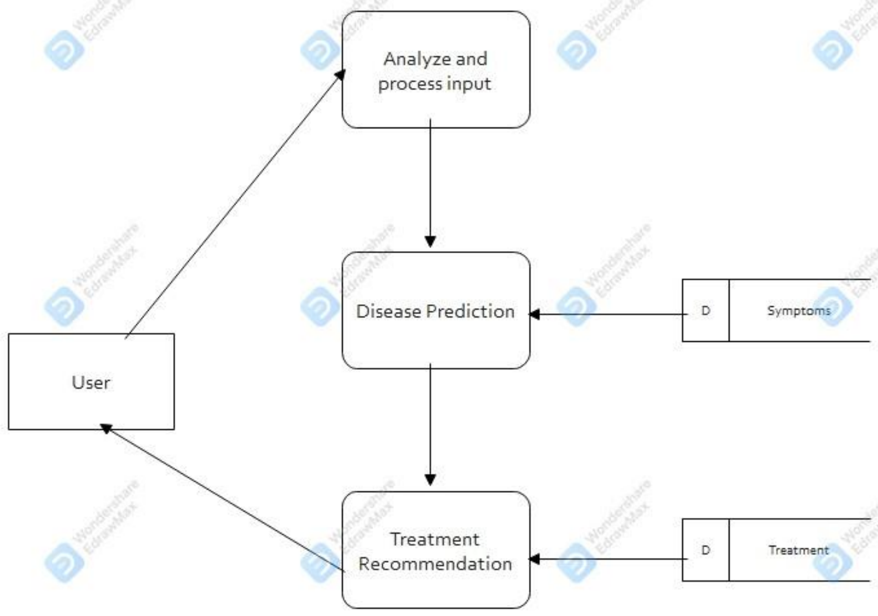


Fig 2.4 Data Flow Diagram

Object Diagram:

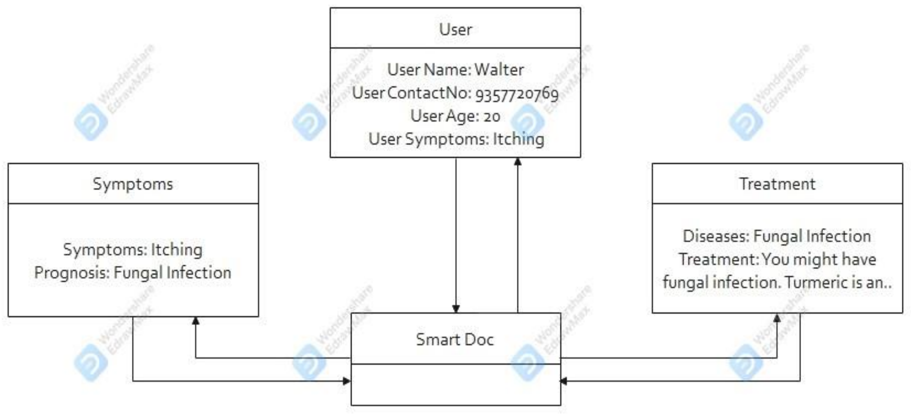

User
User Name: Walter
User ContactNo: 9357720769
User Age: 20
User Symptoms: Itching
Symptoms
Symptoms: Itching
Prognosis: Fungal Infection
Treatment
Diseases: Fungal Infection
Treatment: You might have fungal infection. Turmeric is an..
Smart Doc

Fig 2.5 Object Diagram

2.5 ALGORITHMS

2.4.1 Support Vector Classifier:

Support Vector Classifier is a discriminative classifier i.e. when given a labeled training data, the algorithm tries to find an optimal hyperplane that accurately separates the samples into different categories in hyperspace.

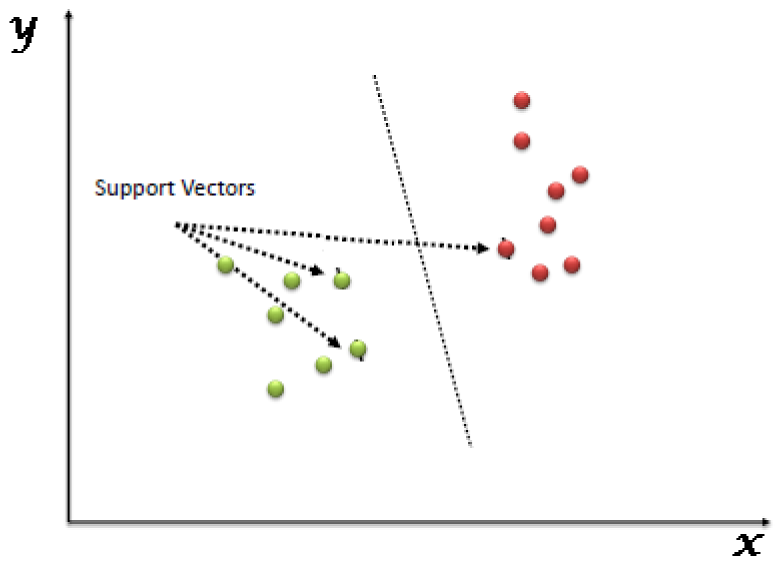


Fig 2.6 Support Vector Classifier

2.4.2 Random Forest Classifier:

Random Forest is an ensemble learning-based supervised machine learning classification algorithm that internally uses multiple decision trees to make the classification. In a random forest classifier, all the internal decision trees are weak learners, the outputs of these weak decision trees are combined i.e. mode of all the predictions is as the final prediction.

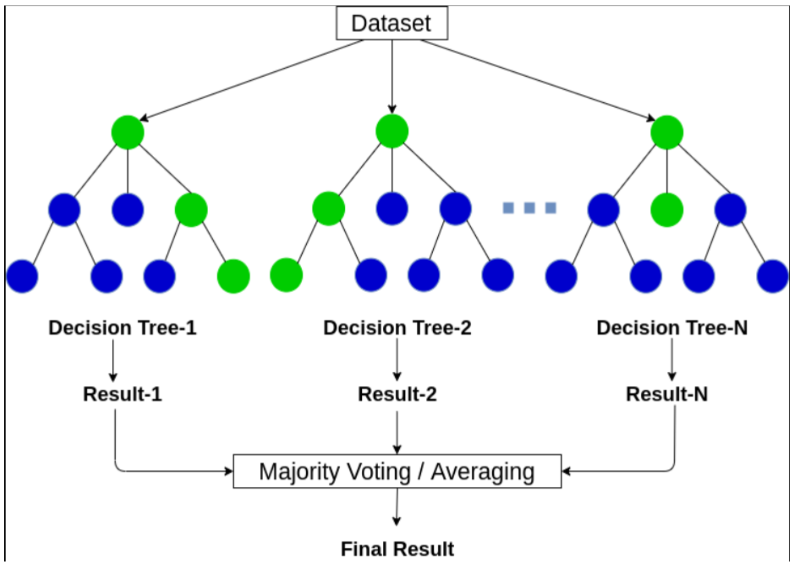
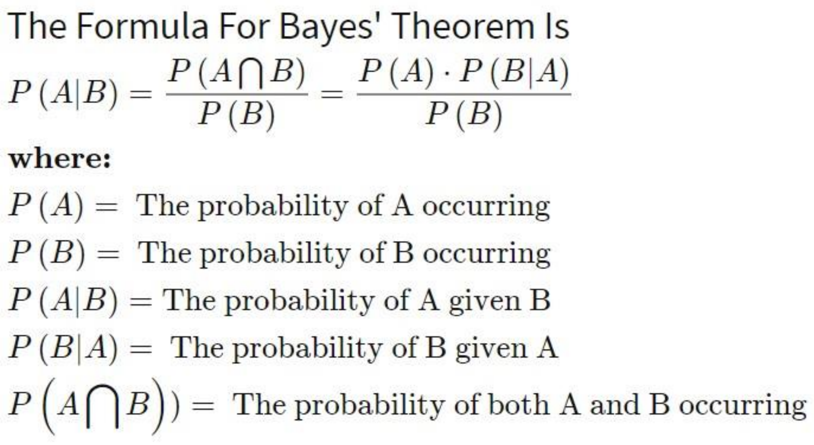


Fig 2.7 Random Forest Classifier

2.4.3 Gaussian Naïve Bayes Classifier:

It is a probabilistic machine learning algorithm that internally uses Bayes Theorem to classify the data points. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.



The Formula For Bayes' Theorem Is

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

$P(B|A) =$ The probability of B given A

$P\left(A \bigcap B\right)) =$ The probability of both A and B occurring

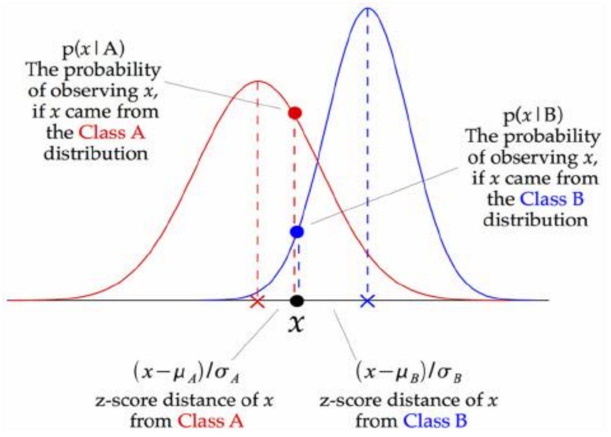
Fig 2.8 Naïve Bayes Theorem

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Fig 2.9 Gaussian Naïve Bayes Classifier

Sometimes assume variance

- is independent of Y (i.e., σi),
- or independent of Xi (i.e., σk)
- or both (i.e., σ)



p(x | A)
The probability of observing x, if x came from the Class A distribution
p(x | B)
The probability of observing x, if x came from the Class B distribution
x
$(x-\mu_A)/\sigma_A$
z-score distance of x from Class A
$(x-\mu_B)/\sigma_B$
z-score distance of x from Class B

Fig 2.10 Illustration of how Gaussian Naïve Bayes Classifier works for each data point

The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class. Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently.

CHAPTER- 3

IMPLEMENTATION

3.1 SOURCE CODE

HTML:

```
# Importing the required modules
from flask import Flask, url_for, render_template, request
from predict import predictDisease
from response import check_all_messages
from flask_mysqldb import MySQL
import yaml

# Creating the Flask app
app=Flask(__name__)

# Configure db
db= yaml.safe_load(open('db.yaml'))
app.config['MYSQL_HOST']= db['mysql_host']
app.config['MYSQL_USER']= db['mysql_user']
app.config['MYSQL_PASSWORD']= db['mysql_password']
app.config['MYSQL_DB']= db['mysql_db']


mysql= MySQL(app)


# Creating the route
@app.route('/')
```

```
# Function to return the required template
def home():
    return render_template('index.html')

is_name = False
is_age = False
is_contact= False
Name=""
Age=""
ContactNumber=""


#Creating the route to communicate with the user
@app.route('/get')
def reply():
    userText=request.args.get('msg')
    global is_name
    global is_age
    global is_contact
    global Name
    global Age
    global ContactNumber
    if(is_name == False):
        Name=userText
        is_name = True
        return "Could you please mention your age?"
    elif(is_age == False):
        Age=userText
        is_age = True
        return "Please Provide your contact number."
```

```
 elif(is_contact == False):
        ContactNumber=userText
        is_contact = True
        # Connecting to MySQL and Inserting user details
        cur=mysql.connection.cursor()
        cur.execute("INSERT INTO users(UserName,Age,ContactNumber) VALUES(%s,
%s, %s)",(Name,Age,ContactNumber))
        mysql.connection.commit()
        cur.close()
        return "Thank you for providing the details. How may I help you?"
    return str(check_all_messages(userText))


if __name__=='__main__':
    app.run(debug=True)
```

CSS:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/*main div*/

.main {
    width: 100vw;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-image: linear-gradient(rgb(44, 50, 93), rgb(30, 31, 53));
}
```

```css
/*Mic Action*/

.microphoneAction {
    color: aliceblue;
    height: 5vh;
    background-color: rgb(40, 41, 63);
    border-radius: 5px;
    display: flex;
    justify-content: center;
    align-items: center;
    text-align: center;
    position: absolute;
    z-index: 20;
    width: 20vw;
    bottom: 4vh;
    display: none;
}

/*stop*/

.stop {
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    bottom: 4.2vh;
    left: 57.5vw;
    cursor: pointer;
    z-index: 20;
    opacity: 0.7;
    display: none;
}
```

```css
/*sub div*/

.sub {
   width: 70vw;
   height: 80vh;
   border-radius: 10px;
   background-color: rgb(5, 12, 50);
   position: absolute;
   overflow-y: auto;
   padding-bottom: 200px;
   overflow-wrap: break-word;
}


/*input tag*/

.chat {
   width: 70vw;
   z-index: 100000;
   position: absolute;
   bottom: 9.4vh;
}


/*mic*/

#talk {
   position: absolute;
   width: 1.9vw;
   height: 5.5vh;
   cursor: pointer;
   opacity: 0.7;
   background-color: rgb(44, 50, 93);
   border-radius: 0px 0px 10px 0px;
}
```

```
/*textbox*/

#send {
   width: 68.1vw;
   height: 5.5vh;
   overflow-y: auto;
   background-color: rgb(44, 50, 93);
   opacity: 0.7;
   border-radius: 0px 0px 0px 10px;
   font-size: 1.2rem;
   outline: none;
   border: none;
   color: rgb(240, 248, 255);
   resize: none;
}


/*message*/

.msg {
   width: 34vw;
   height: auto;
   background-image: linear-gradient(rgb(255, 157, 157), rgb(128, 79, 79));
   opacity: 0.5;
   color: rgb(5, 12, 30);
   margin-top: 2vh;
   padding: 7px;
   border-radius: 12px;
}


/*bot message*/

.msg.bot {
   margin-left: 2vw;
   float: left;
}


/*user message*/

.msg.user {
   margin-right: 2vw;
   float: right;
}
```

Javascript:

```
var number = 0;

//Function that replies back
function botAns(userInput) {
    //split the input into an array of strings whenever a blank space is encountered
    const arr = userInput.split(" ");

    //loop through each element of the array and capitalize the first letter
    for (var i = 0; i < arr.length; i++) {
        arr[i] = arr[i].charAt(0).toUpperCase() + arr[i].slice(1);

    }

    //Join all the elements of the array back into a string using a blankspace as a separator
    const modInput = arr.join(" ");

    //div element which holds the input
    var user = document.createElement('div');
    var bot = document.createElement('div');

    //Assigning the class
    user.className = 'msg user';
    bot.className = 'msg bot';

    //TextNode to display the input given by user
    var userEntry = document.createTextNode(userInput);
    document.getElementById('sub').appendChild(user);
    let x = document.getElementsByClassName('user');
    x[number].appendChild(userEntry);
    number += 1;
//Passing the input entered by user to Flask app
    $.get('/get', { msg: modInput }).done(function(data) {
        //TextNode to display the reply given by bot
        readOutLoud(data);
        botRpl = document.createTextNode(data);
        document.getElementById('sub').appendChild(bot);
        let y = document.getElementsByClassName('bot');
        y[number].appendChild(botRpl);

        //Clearing the textbox after sending the message
        document.getElementById('send').value = '';
    })
}
//Function that reads out loud
function readOutLoud(speak) {
    let speech = new SpeechSynthesisUtterance();

    speech.lang = "en-US";
    speech.text = speak;
    speech.volume = 1;
    speech.rate = 1;
    speech.pitch = 1;

    window.speechSynthesis.speak(speech);
}
```

```
//Function that Gets the input given by user
function myKeypress(e) {
    if (window.event) {
        if (e.keyCode == 13) {

            //Auto Scroll
            $('#sub').scrollTop($('#sub').scrollTop() + 100);

            var entry = document.getElementById('send').value;
            //Passing the data given by user to a function that will reply back
            botAns(entry)
        }
    }
}

//Making the microphone work
function myClick(e) {
    document.getElementById('microphoneAction').style.display = 'flex';
    document.getElementById('stop').style.display = 'flex';

    //Speech recognition object
    var SpeechRecognition = SpeechRecognition || webkitSpeechRecognition;
    var recognition = new SpeechRecognition();

    //Runs when the speech recognition starts
    recognition.onstart = function() {
        document.getElementById('microphoneAction').innerHTML = "Listening..Please
Speak!";
    };
//Runs When user is done speaking
   recognition.onspeechend = function() {
      recognition.stop();
      document.getElementById('microphoneAction').style.display = 'none';
      document.getElementById('stop').style.display = 'none';
   }

   //Runs when the speech recognition returns result
   recognition.onresult = function(event) {
      var userMicrophone = event.results[0][0].transcript;
      //Getting the ans from the bot
      botAns(userMicrophone);

   };

   // Start recognition
   recognition.start();

   //Stop recognition
   document.getElementById('stop').addEventListener('click', (e) => {
      recognition.stop();
      document.getElementById('microphoneAction').style.display = 'none';
      document.getElementById('stop').style.display = 'none';

   });
}
```

Python- Response:

```
# Importing libraries
import re
import random
from predict import predictDisease

# Function to check the probability of the message
def message_probability(user_message, recognised_words, single_response=False,
required_words=[]):
    message_certainty = 0
    has_required_words = True

    # Counts how many words are present in each predefined message
    for word in user_message:
        if word in recognised_words:
            message_certainty += 1

    # Calculates the percent of recognised words in a user message
    percentage = float(message_certainty) / float(len(recognised_words))

    # Checks that the required words are in the string
    for word in required_words:
        if word not in user_message:
            has_required_words = False
            break

    # Must either have the required words, or be a single response
    if has_required_words or single_response:
        return int(percentage * 100)
    else:
        return 0
```

```
# Function to return the answer based on user input
def check_all_messages(user_input):
    message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
    highest_prob_list = {}


    # Simplifies response creation / adds it to the dict
    def response(bot_response, list_of_words, single_response=False, required_words=[]):
        nonlocal highest_prob_list
        highest_prob_list[bot_response] = message_probability(message, list_of_words,
single_response, required_words)


    # Responses -------------------------------------------------------------------------------------------
--------------
    response(hello(),['hello','hi','hey','hai','smart','doc','hei','hay'], single_response=True)

response(greetings(),['morning','mrng','afternoon','noon','nun','evening','eve','greetings'],si
ngle_response=True)
    response(bye(),['bye','goodbye','bubye','tata'], single_response=True)
    response(help(),['please','help','pls'], single_response=True)
    response(how(),['how','are','you','doing'], required_words=['how'])
    response(good(),['fine','good','great','better','awesome','nice'], single_response=True)
    response(thanks(),['thank','thanks'], single_response=True)
    response(yeah(),['yeah','sure','yes','definitely','yup', 'ok','okie'], single_response=True)
    response(pain(),['i','have','got','not','feeling','paining','sick','tired'],
single_response=True)
    response(book_appointment(),['book','appointment',
'doctor','schedule'],single_response=True)
```

```
# Functions to respond back------------------------------------------------------
def hello():
   hello=['Hi, How can I help you today?',
         'Hello there, How can I be of service?',
         'Hi, How can I help you?',
         'Hello, How can I help?',
         'Hey, How can I help?',
         'Hi. It\'s good to hear from you. How can I help?'
        ][
   random.randrange(6)]
   return hello
def greetings():
   greetings=['Greetings of the day! How are you doing?',
           'Nice to see you!',
           'Greetings of the day!',
           'Hi. It\'s good to hear from you. How can I help?',
           'Have a great day!',
           'Hey, How can I help?'
          ][
   random.randrange(6)]
   return greetings
def bye():
   bye=['Take care, See you!',
       'Have a nice day!',
       'Take care, bye!',
       'bye',
       'ta-ta',
       'goodbye'
      ][
   random.randrange(6)]
   return bye
```

```
def help():
   help=['I\'m here to help you.',
        'I\'ll do my best.',
        'I\'ll do all I can.',
        'I\'m glad to help you.',
        'I\'m here to listen.',
        'I would love to help you.'
       ][
   random.randrange(6)]
   return help
def how():
   how=['I\'m doing fine, and you?',
       'I\'m fine. You\'re very kind to ask, especially in these tempestous times.',
       'I\'m splendid, Thank you for asking.',
       'Great, thanks. What can I do for you?',
       'I\'m great. Thank you for asking.',
       'Awesome! What about you?'
      ][
   random.randrange(6)]
   return how
def good():
   good=['It\'s awesome being able to help.',
        'Great!',
        'It\'s good to hear.',
        'Cool.',
        'Cool, Is there anything else I can do?',
        'I\'m happy you\'re happy.'
       ][
   random.randrange(6)]
   return good
```

```
def thanks():
    thanks=['You\'re very welcome!',
            'I\'m honoured to serve.',
            'No worries, I\'m here to help.',
            'I\'m here to help.',
            'You\'re the best, I love helping you.',
            'Just doing my job.'
           ][
    random.randrange(6)]
    return thanks
def yeah():
    yeah=['Okie..',
          'Nice',
          'Ok',
          'Okay',
          'Cool',
          'Right!'
        ][
    random.randrange(6)]
    return yeah
def pain():
    pain=['That does not sound good. Could you please mention your symptoms?',
          'I\'m sorry to hear that. Could you please mention your symptoms?',
          'Could you please mention your symptoms?',
          'I\'m sorry. Could you please mention your symptoms?',
          'I can understand. Could you please mention your symptoms?',
          'Okie, I get it. Could you please mention your symptoms?'
        ][
    random.randrange(6)]
    return pain
```

```
def book_appointment():
    website="https://www.apollo247.com/specialties"
    return "Please go to this link to book an appointment: "+website


def symptoms(rawData):
    ans=predictDisease(rawData)
    return ans
```

Python- Prediction:

```
# Importing libraries
import numpy as np
import pandas as pd
from scipy.stats import mode
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import random

# Reading the training data
data_path="Diseases.csv"
data=pd.read_csv(data_path)

# Encoding the target value (prognosis column) into numerical value using LabelEncoder
encoder = LabelEncoder()
data["prognosis"] = encoder.fit_transform(data["prognosis"])
```

```
# Training the models
X = data.iloc[:,:-1]
Y= data.iloc[:, -1]
X_train, X_test, y_train, y_test =train_test_split(X, Y, test_size = 0.2, random_state = 24)
# print(f"Train: {X_train.shape}, {y_train.shape}")
# print(f"Test: {X_test.shape}, {y_test.shape}")

# Training and testing SVM Classifier
svm_model = SVC()
svm_model.fit(X_train, y_train)
preds = svm_model.predict(X_test)
# print(f"Accuracy on train data by SVM Classifier: {accuracy_score(y_train,
svm_model.predict(X_train))*100}")
# print(f"Accuracy on test data by SVM Classifier: {accuracy_score(y_test,
preds)*100}")

# Training and testing Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
preds = nb_model.predict(X_test)
# print(f"Accuracy on train data by Naive Bayes Classifier: {accuracy_score(y_train,
nb_model.predict(X_train))*100}")
# print(f"Accuracy on test data by Naive Bayes Classifier: {accuracy_score(y_test,
preds)*100}")

# Training and testing Random Forest Classifier
rf_model = RandomForestClassifier(random_state=18)
rf_model.fit(X_train, y_train)
preds = rf_model.predict(X_test)
# print(f"Accuracy on train data by Random Forest Classifier:
{accuracy_score(y_train, rf_model.predict(X_train))*100}")
# print(f"Accuracy on test data by Random Forest Classifier:
{accuracy_score(y_test, preds)*100}")
```

```
# Training the models on whole data
final_svm_model = SVC()
final_nb_model = GaussianNB()
final_rf_model = RandomForestClassifier(random_state=18)
final_svm_model.fit(X, Y)
final_nb_model.fit(X, Y)
final_rf_model.fit(X, Y)

symptoms = X.columns.values

# Creating a symptom index dictionary to encode the input symptoms into numerical
form
symptom_index = {}
for index, value in enumerate(symptoms):
symptom = " ".join([i.capitalize() for i in value.split("_")])
symptom_index[symptom] = index

data_dict = {
"symptom_index":symptom_index,
"predictions_classes":encoder.classes_
}

# Defining the Function Input: string containing symptoms separated by commmas
Output: Generated predictions by models
def predictDisease(symptoms):
    symptoms = symptoms.split(", ")
```

```
# creating input data for the models
    input_data = [0] * len(data_dict["symptom_index"])
    try:
        for symptom in symptoms:
            index = data_dict["symptom_index"][symptom]
            input_data[index] = 1
        # reshaping the input data and converting it
        # into suitable format for model predictions
        input_data = np.array(input_data).reshape(1,-1)

        # generating individual outputs
        rf_prediction =
data_dict["predictions_classes"][final_rf_model.predict(input_data)[0]]
        nb_prediction =
data_dict["predictions_classes"][final_nb_model.predict(input_data)[0]]
        svm_prediction =
data_dict["predictions_classes"][final_svm_model.predict(input_data)[0]]

        # making final prediction by taking mode of all predictions
        final_prediction = mode([rf_prediction, nb_prediction, svm_prediction])[0][0]

        remedy=cure(final_prediction)
        return "You might have "+final_prediction+ ". "+remedy
```

```
except KeyError:
        rpl=["Could you please re-phrase that? ",
            "I’m sorry, I don’t understand.",
            "Sorry, I didn’t get that.",
            "I can’t make head nor tail of what you’re saying.",
            "What does that mean?",
            "Can you try saying that again in a different way? I don't understand.",
            ][
        random.randrange(6)]
        return rpl

# Function to return an approriate treatment for the predicted disease
def cure(remedy):
    treatment = pd.read_csv("Treatment.csv",encoding='latin1')
    for rows in treatment.iterrows():
        medication=treatment.loc[treatment['Prognosis'] == remedy]
        pd.options.display.max_colwidth = 1000
        return medication['Treatment'].to_string(index=False)
```

SQL:

```
create database smartdoc;
use smartdoc
create table users(Username varchar(20), Age varchar(3), ContactNumber varchar(15));
```

Python- Flask:

```
# Importing the required modules
from flask import Flask, url_for, render_template, request
from predict import predictDisease
from response import check_all_messages
from flask_mysqldb import MySQL
import yaml

# Creating the Flask app
app=Flask(__name__)

# Configure db
db= yaml.safe_load(open('db.yaml'))
app.config['MYSQL_HOST']= db['mysql_host']
app.config['MYSQL_USER']= db['mysql_user']
app.config['MYSQL_PASSWORD']= db['mysql_password']
app.config['MYSQL_DB']= db['mysql_db']


mysql= MySQL(app)


# Creating the route
@app.route('/')
```

```
# Function to return the required template
def home():
    return render_template('index.html')
is_name = False
is_age = False
is_contact= False
Name=""
Age=""
ContactNumber=""

#Creating the route to communicate with the user
@app.route('/get')
def reply():
    userText=request.args.get('msg')
    global is_name
    global is_age
    global is_contact
    global Name
    global Age
    global ContactNumber
    if(is_name == False):
        Name=userText
        is_name = True
        return "Could you please mention your age?"
    elif(is_age == False):
        Age=userText
        is_age = True
        return "Please Provide your contact number."
    elif(is_contact == False):
        ContactNumber=userText
        is_contact = True
```

```
# Connecting to MySQL and Inserting user details
        cur=mysql.connection.cursor()
        cur.execute("INSERT INTO users(UserName,Age,ContactNumber) VALUES(%s,
%s, %s)",(Name,Age,ContactNumber))
        mysql.connection.commit()
        cur.close()
        return "Thank you for providing the details. How may I help you?"
    return str(check_all_messages(userText))


if __name__=='__main__':
    app.run(debug=True)
```

3.2 OUTPUT

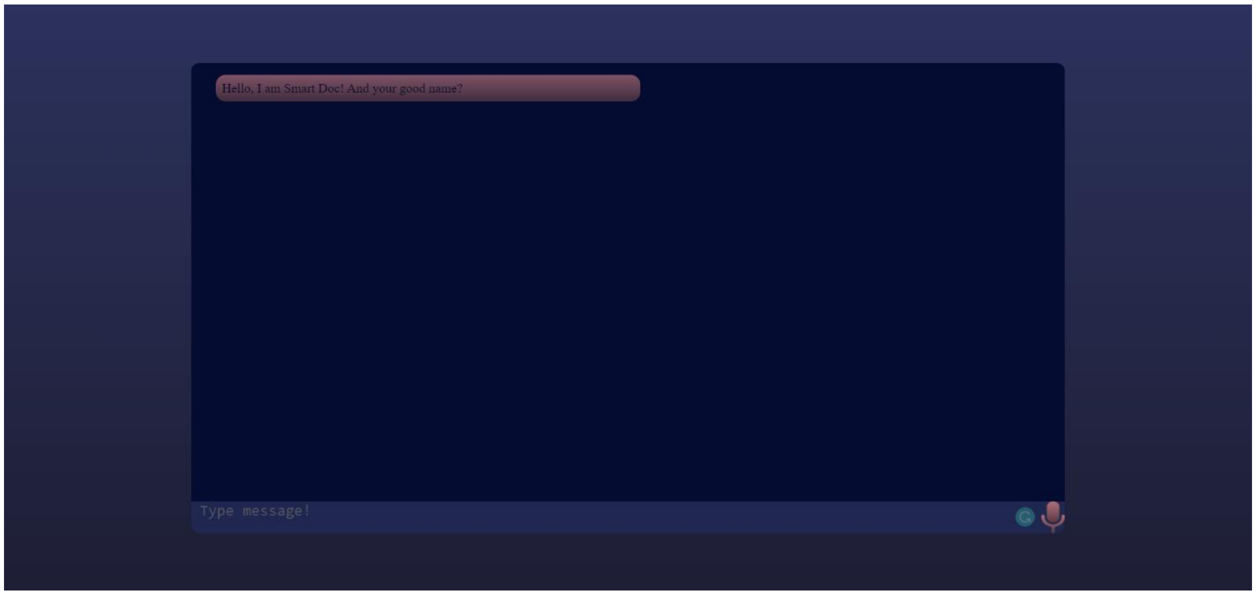
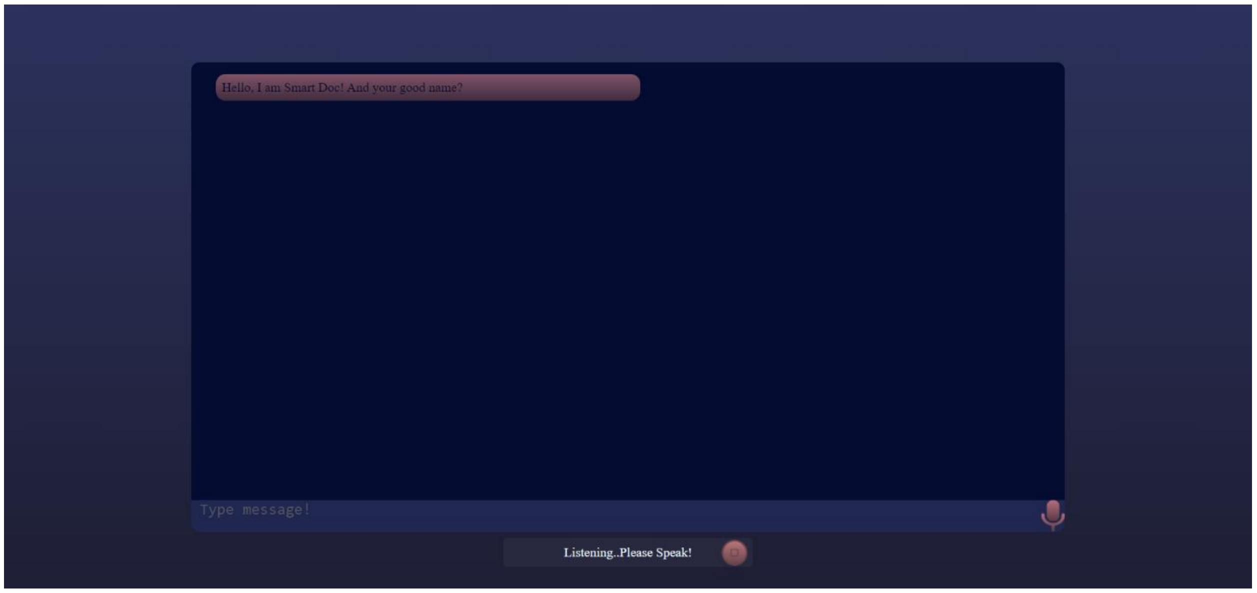

Hello, I am Smart Doc! And your good name?

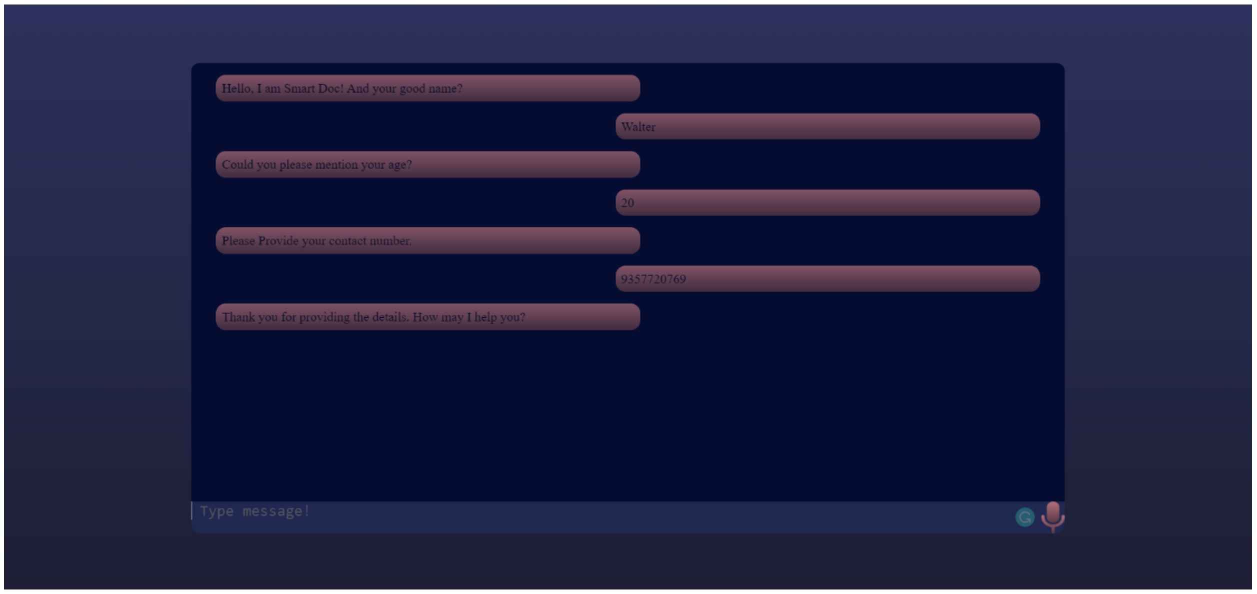
Fig 3.1 Smart Doc


Hello, I am Smart Doc! And your good name?

Type message!

Listening..Please Speak!

Fig 3.2 Mic Access


Hello, I am Smart Doc! And your good name?

Walter

Could you please mention your age?

20

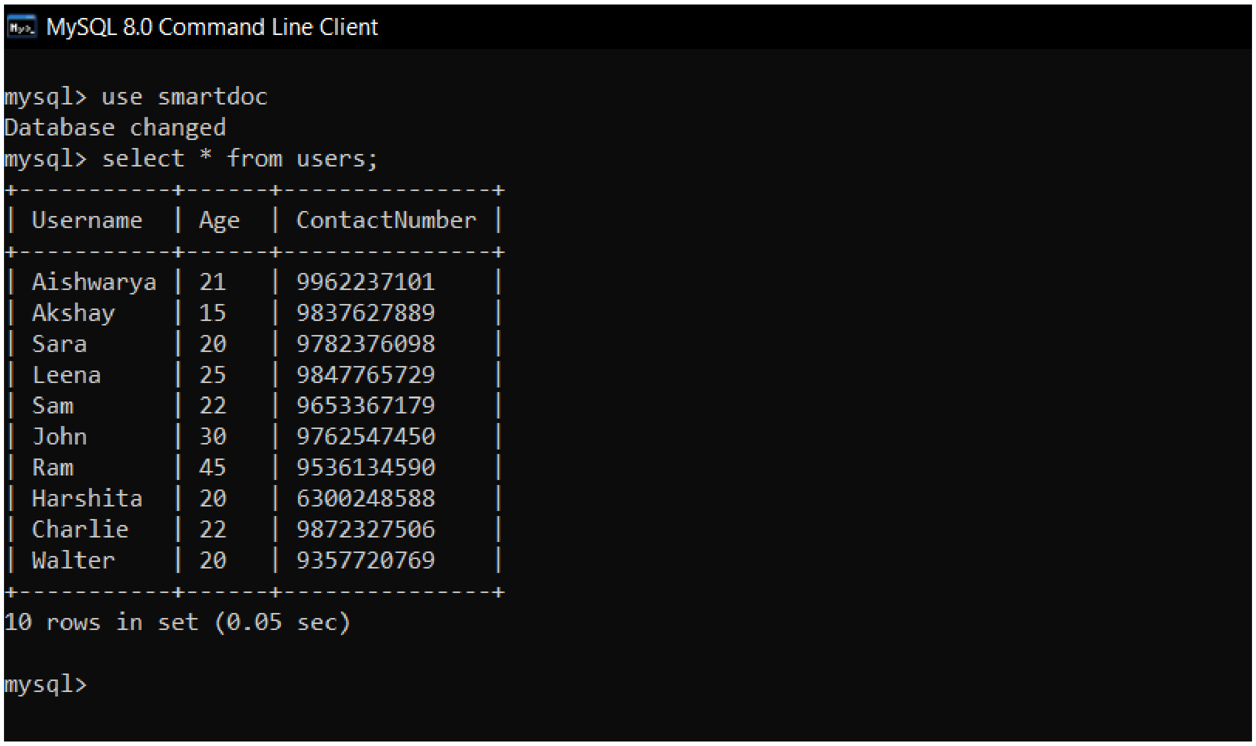
Please Provide your contact number.

9357720769

Thank you for providing the details. How may I help you?

Type message!

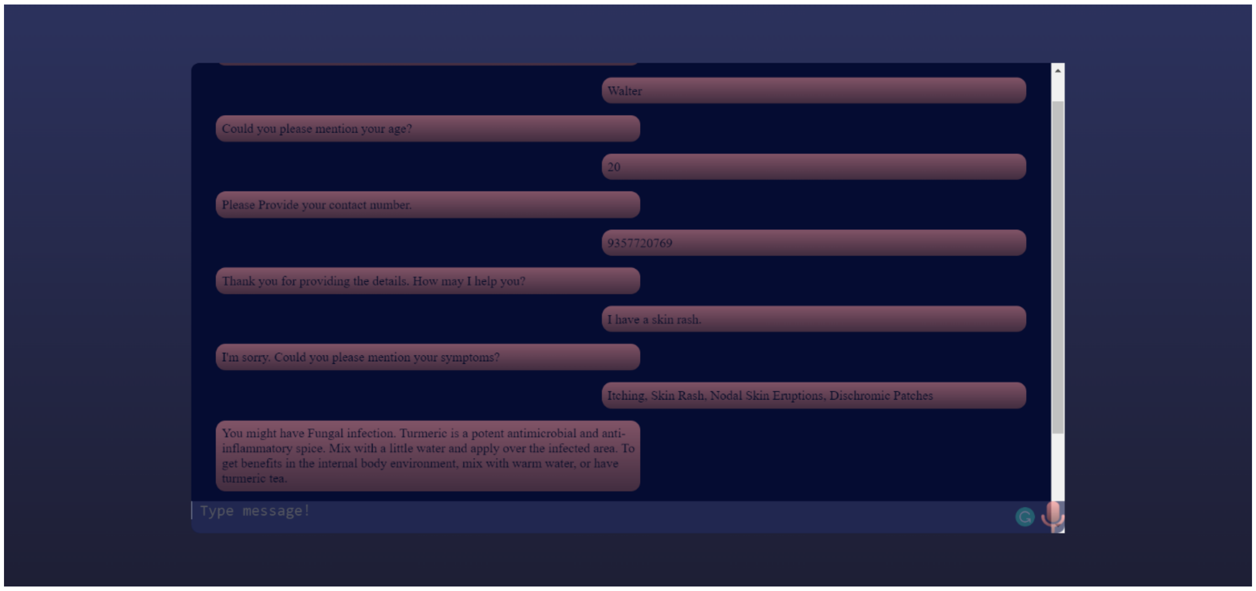
Fig 3.3 Getting User Details


MySQL 8.0 Command Line Client

```
mysql> use smartdoc
Database changed
mysql> select * from users;
+-----------+------+---------------+
| Username  | Age  | ContactNumber |
+-----------+------+---------------+
| Aishwarya | 21   | 9962237101    |
| Akshay    | 15   | 9837627889    |
| Sara      | 20   | 9782376098    |
| Leena     | 25   | 9847765729    |
| Sam       | 22   | 9653367179    |
| John      | 30   | 9762547450    |
| Ram       | 45   | 9536134590    |
| Harshita  | 20   | 6300248588    |
| Charlie   | 22   | 9872327506    |
| Walter    | 20   | 9357720769    |
+-----------+------+---------------+
10 rows in set (0.05 sec)

mysql>
```

Fig 3.4 User Table


Walter

Could you please mention your age?

20

Please Provide your contact number.

9357720769

Thank you for providing the details. How may I help you?

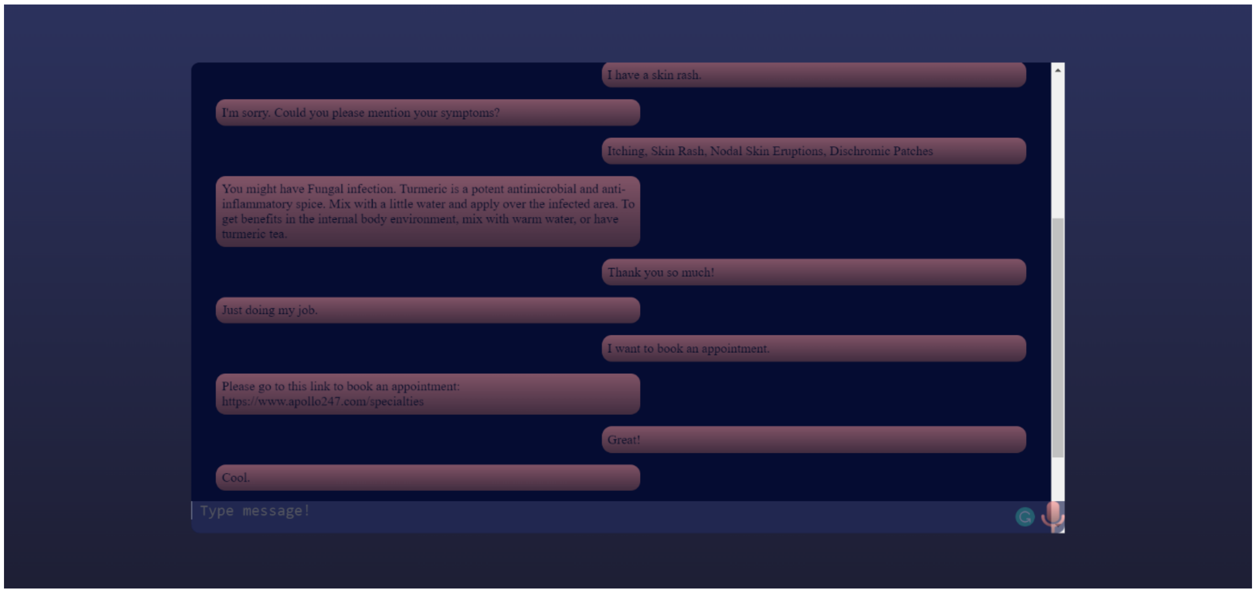
I have a skin rash.

I'm sorry. Could you please mention your symptoms?

Itching, Skin Rash, Nodal Skin Eruptions, Dischromic Patches

You might have Fungal infection. Turmeric is a potent antimicrobial and anti-inflammatory spice. Mix with a little water and apply over the infected area. To get benefits in the internal body environment, mix with warm water, or have turmeric tea.

Fig 3.5 Disease Prediction and Treatment



I have a skin rash.

I'm sorry. Could you please mention your symptoms?

Itching, Skin Rash, Nodal Skin Eruptions, Dischromic Patches

You might have Fungal infection. Turmeric is a potent antimicrobial and anti-inflammatory spice. Mix with a little water and apply over the infected area. To get benefits in the internal body environment, mix with warm water, or have turmeric tea.

Thank you so much!

Just doing my job.

I want to book an appointment.

Please go to this link to book an appointment:
https://www.apollo247.com/specialties

Great!

Cool.

Fig 3.6 Schedule Appointment

CHAPTER- 4
CONCLUSION

4.1 CONCLUSION

A Chatbot is never tired; it is never short of patience for a patient! A Chatbot hears your patients, analyzes their response and then carefully delivers a tailor made reply. Patient satisfaction in these type of encounters is maximum as they get what they want without any diversions or delays. This chatbot will help you get instant resolutions with hands on experience for the stated issues. It makes your work easy and produce adequate results as they are automated.

4.2 FUTURE SCOPE

Since the system aims at improving medical assistance, subsequent improvements to the dataset and to the system are required which is facilitated by the thorough evaluation in order to enable scalable implementation. Thus enabling evaluation and analysis by various domain experts to iteratively improve the dataset as well as the system. By adding new functionalities, such as the management of medical records, and the automatic suggestion of food and physical activity to perform based on the user's health conditions will enhance the scope of the project in future.

CHAPTER- 5
REFERENCES

[1] Nikita Vijay Shinde, Aniket Akhade, Pranali Bagad, Harshit Bhavsar, Dr. S.K.Wagh, Prof. Amol Kamble, “Healthcare Chatbot System using Artificial Intelligence” 2021, IEEE.

[2] Resmi Ramakrishnan, Pillai Amal Manoj, Krishna Sreejish, Neetisha Uniyal, “Meddoc-The AI Doctor”, 2021, IRJET.

[3] Amela Softić, Jasmina Baraković Husić, Aida Softić, Sabina Baraković, Health Chatbot: Design, Implementation, Acceptance and Usage Motivation”, 2021, IEEE.

[4] Muse Mohamud Mohamed, Professor Wang Zhuopeng, “Artificial Intelligence Health Care Chatbot System”, 2020, IJARCCE.

[5] Sagar Badlani, Tanvi Aditya, Meet Dave, Sheetal Chaudhari, “Multilingual Healthcare Chatbot Using Machine Learning”,2021, IEEE.

[6] Prakhar Srivastava, Nishant Singh, “Automatized Medical Chatbot (Medibot)”, 2020, IEEE.

[7] Ms Menal Dahiya, “A Tool of Conversation: Chatbot”, 2017, IJCSE.

[8] Mrs. Rashmi Dharwadkar, Dr.Mrs. Neeta A. Deshpande, “A Medical ChatBot”, 2018, IJCTT.

[9] Roop Chandrika Mallela, Reddy Lakshmi Bhavani, B. Ankayarkanni, “Disease Prediction Using Machine Learning Techniques”, 2021, IEEE.

[10] Marco Polignano, Fedelucio Narducci, Andrea Iovine, Cataldo Musto, Marco De Gemmis, Giovanni Semeraro, “HealthAssistantBot: A Personal Health Assistant for the Italian Language, 2020, IEEE.

[11] Abdullah Faiz Ur Rahman Khilji, Sahinur Rahman Laskar, Partha Pakray, Rabiah Abdul Kadir, Maya Silvi Lydia, Sivaji Bandyopadhyay, “HealFavor: Dataset and A Prototype System for Healthcare ChatBot”, 2020, IEEE.

[12] Gauri Tawde, Yash Choksi, Roshan Singh, Krishna Samdani, “A Study on Machine Learning enabled IoT devices for Medical Assistance”, 2019, IEEE.

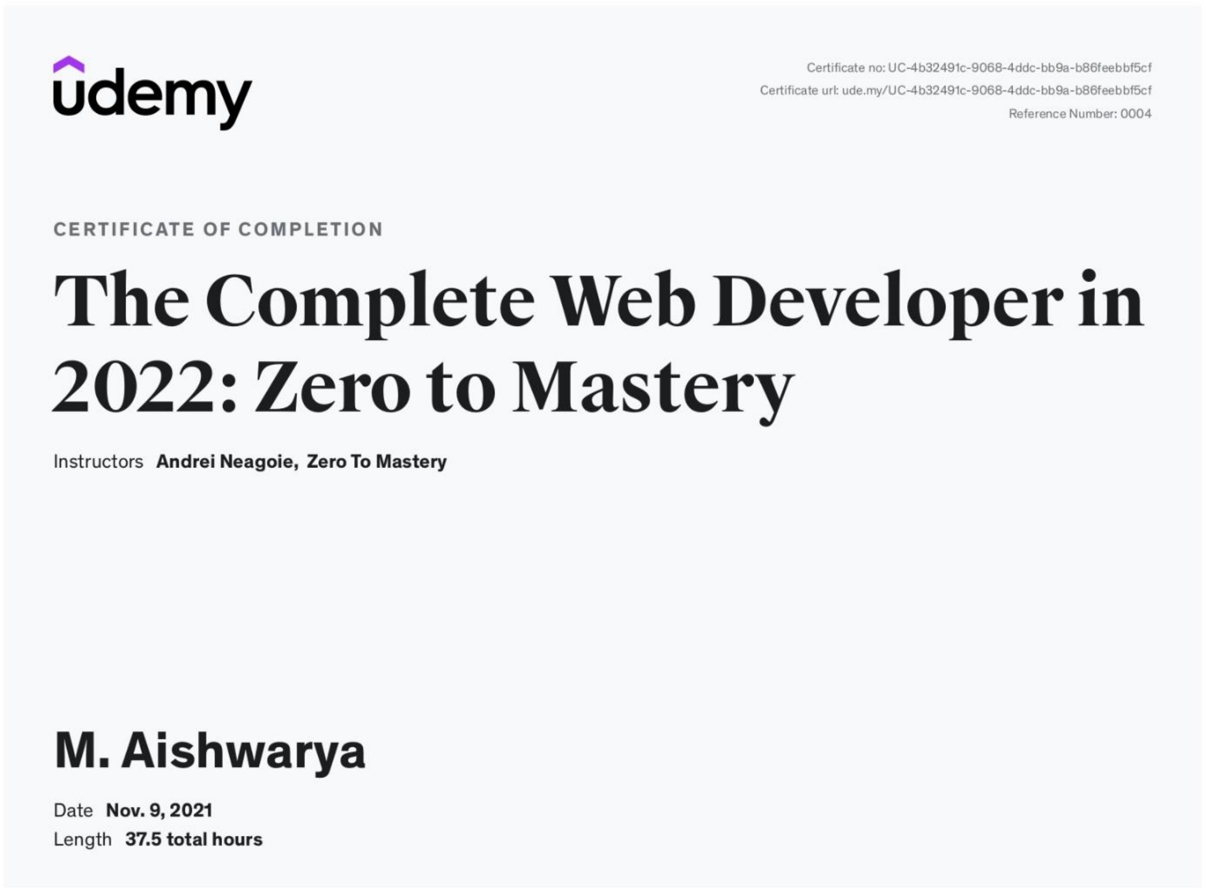
[13] Dr. V Sheeja Kumari, Beksy S George, Nisha Varghese, Sherin Mary Mathew, Telma Elza Ninan, “Humicare- A Chatbot For Healthcare System”, 2021, IRJET.

[14] Hiba Hussain, Komal Aswani, Mahima Gupta, Dr. G.T.Thampi, “Implementation of Disease Prediction Chatbot and Report Analyzer using the Concepts of NLP, Machine Learning and OCR”, 2020, IRJET.

[15] Shadab Adam Pattekari, Asma Parveen, “Prediction System For Heart Disease Using Naïve Bayes”, 2012, IJCMS.

[16] Ruyi Wang, Jiankun Wang, Yuan Liao, Jinyu Wang, “Supervised Machine Learning Chatbots for Perinatal Mental Healthcare”, 2020, IEEE.

UDEMY COURSE COMPLETION CERTIFICATE



udemy

Certificate no: UC-4b32491c-9068-4ddc-bb9a-b86feebbf5cf
Certificate url: ude.my/UC-4b32491c-9068-4ddc-bb9a-b86feebbf5cf
Reference Number: 0004

CERTIFICATE OF COMPLETION

The Complete Web Developer in 2022: Zero to Mastery

Instructors **Andrei Neagoie, Zero To Mastery**

M. Aishwarya

Date **Nov. 9, 2021**
Length **37.5 total hours**