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

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

M. Aishwarya -RA1811003040117

Department : B.Tech, CSE
Semester/Year : VIIth / Ivth
SEC : D

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. With 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 the 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 the 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 need healthcare suggestions before visiting the hospital. The chatbot system works on the inputs provided by the user (patient) and answers it accordingly. The functionalities of Smart Doc include primary healthcare services where a normal person with a mobile can be able to have an 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 the healthcare costs and improve accessibility to medical knowledge even for people living in rural areas while enhancing their lives at the same time.

Introduction

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.

Existing System and Proposed System

Many of the existing systems have live chats through texts and some limitation such as there is no instant response given to the patients they have to wait for experts acknowledgement for a long time. Some of the processes may charge amount to live chat or telephony communication.

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.

Research Gap

Research is being carried out in the development of standard tedious chatbots to be active and responsive and carry out the communication in a normal/natural language. Based on the gathered data as well as the information fed to its smart algorithm, interprets the user's symptoms and recommends a diagnosis. The patient can easily access the chatbot and saves time and money. They can get an accurate and quick response.

Literature Survey

Divya Madhu [1] proposed an idea in which the AI can predict the diseases based on the symptoms and give the list of available treatments If a person‘s body is analyzed periodically, it is possible to predict any possible problem even before they start to cause any damage to the body. Some Challenges are research and implementation costs, and government regulations for the successful implementation of personalized medicine, they are not mentioned in the paper.

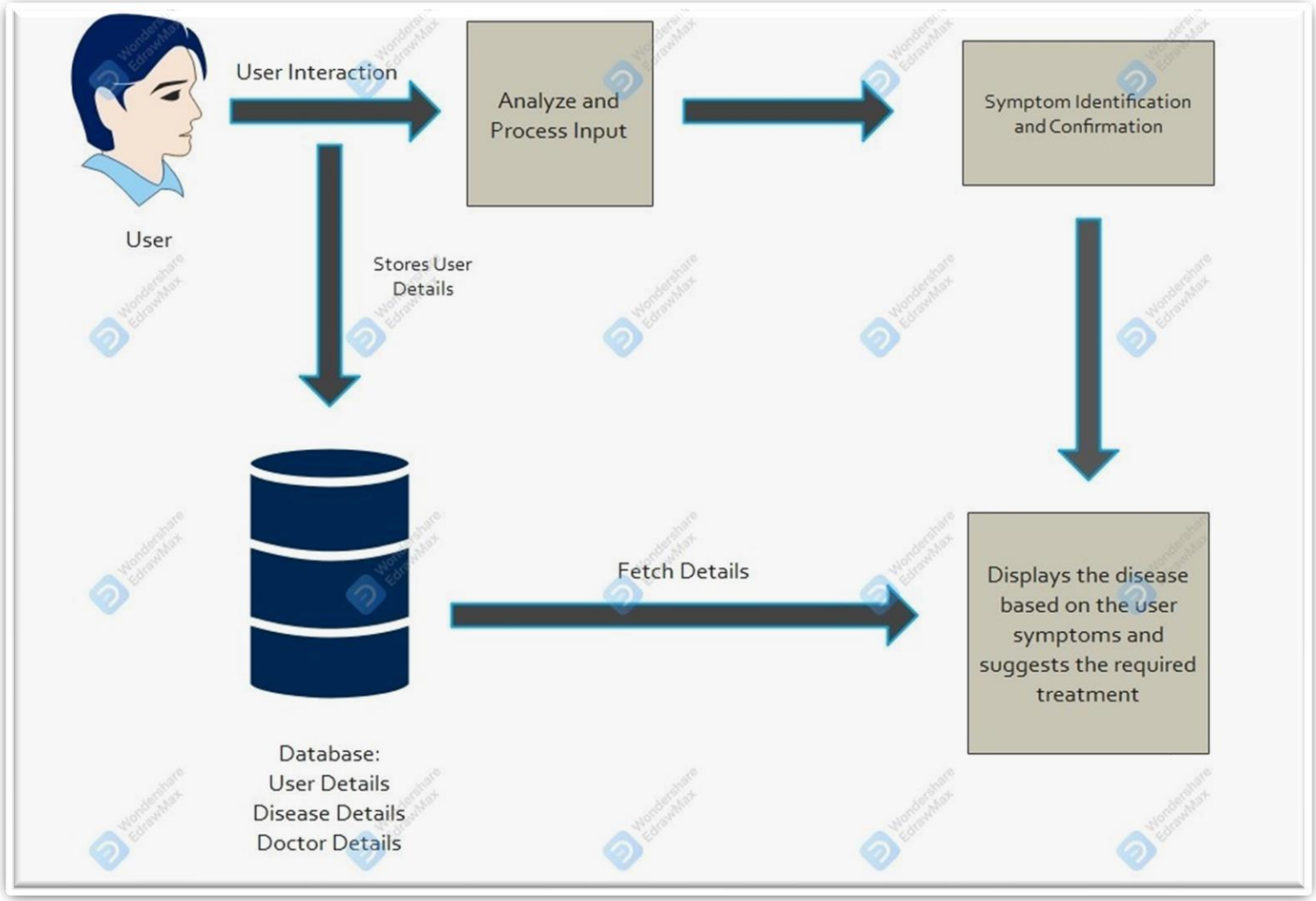
Hameedullah Kazi [2] describes the development of a chatbot for medical students, that is based on the open source AIML based Chatter bean. The AIML based chatbot is customized to convert natural language queries into relevant SQL queries. A total of 97 question samples were collected and then those questions were divided into categories depending on the type of question. According to the number of questions in each category the resultant categories were ranked. Questions were based on queries, where 47% are of posed questions

Saurav Kumar Mishra [3] says that the chatbot will act as a virtual doctor and makes possible for the patient to interact with virtual doctor. Natural language processing and pattern matching algorithm for the development of this chatbot. It is developed using the python Language. Based on the survey given it is found that the no of correct answer given by the chatbot is 80% and incorrect/ambiguous answer given is 20%.From this survey of chatbot and analysis of result suggested that this software can be used for teaching and as a virtual doctor for awareness and primary care.

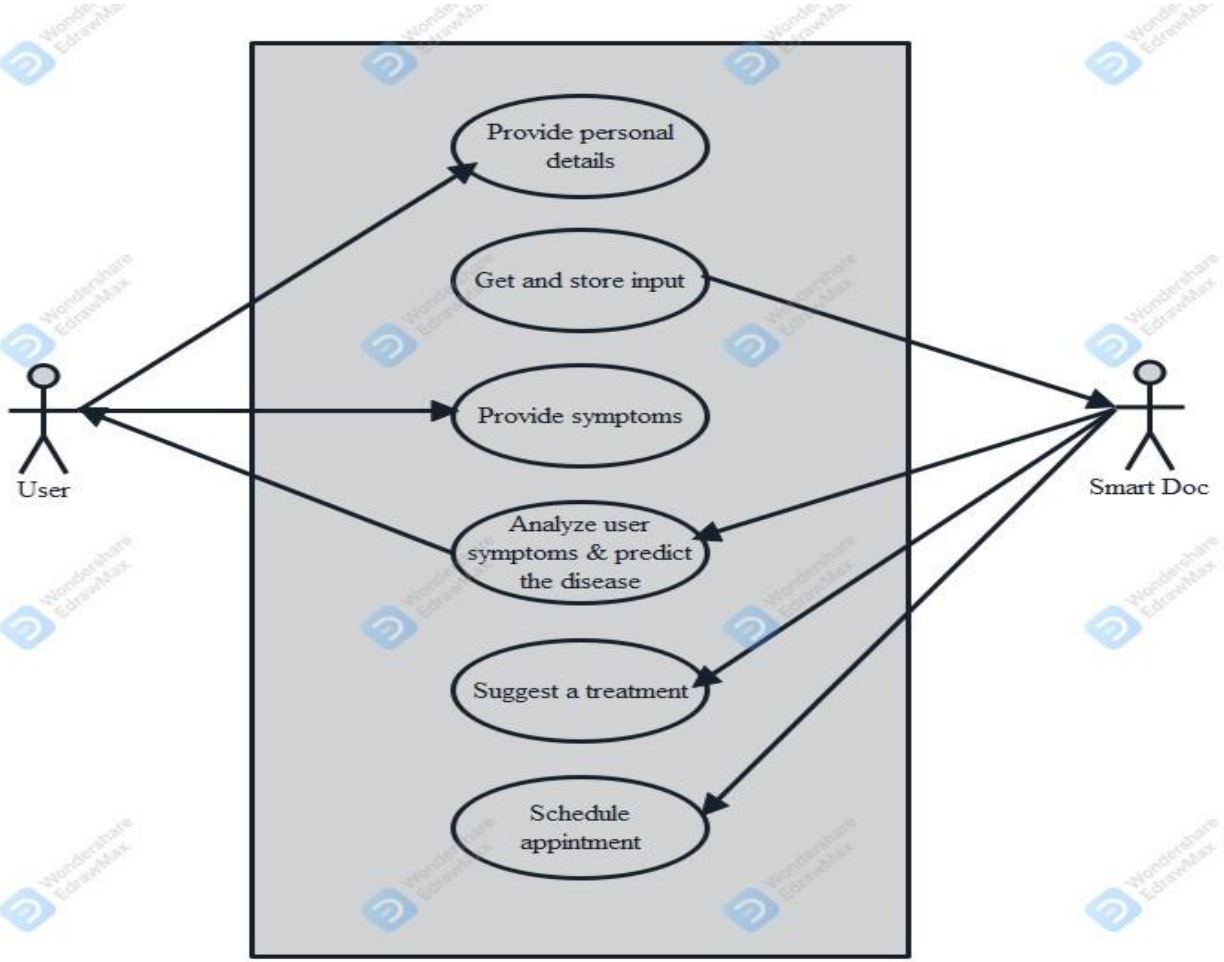
Tools and Technology

- The Python
- Flask
- HTML5
- CSS5
- Java script
- MySQL
- Visual Studio Code

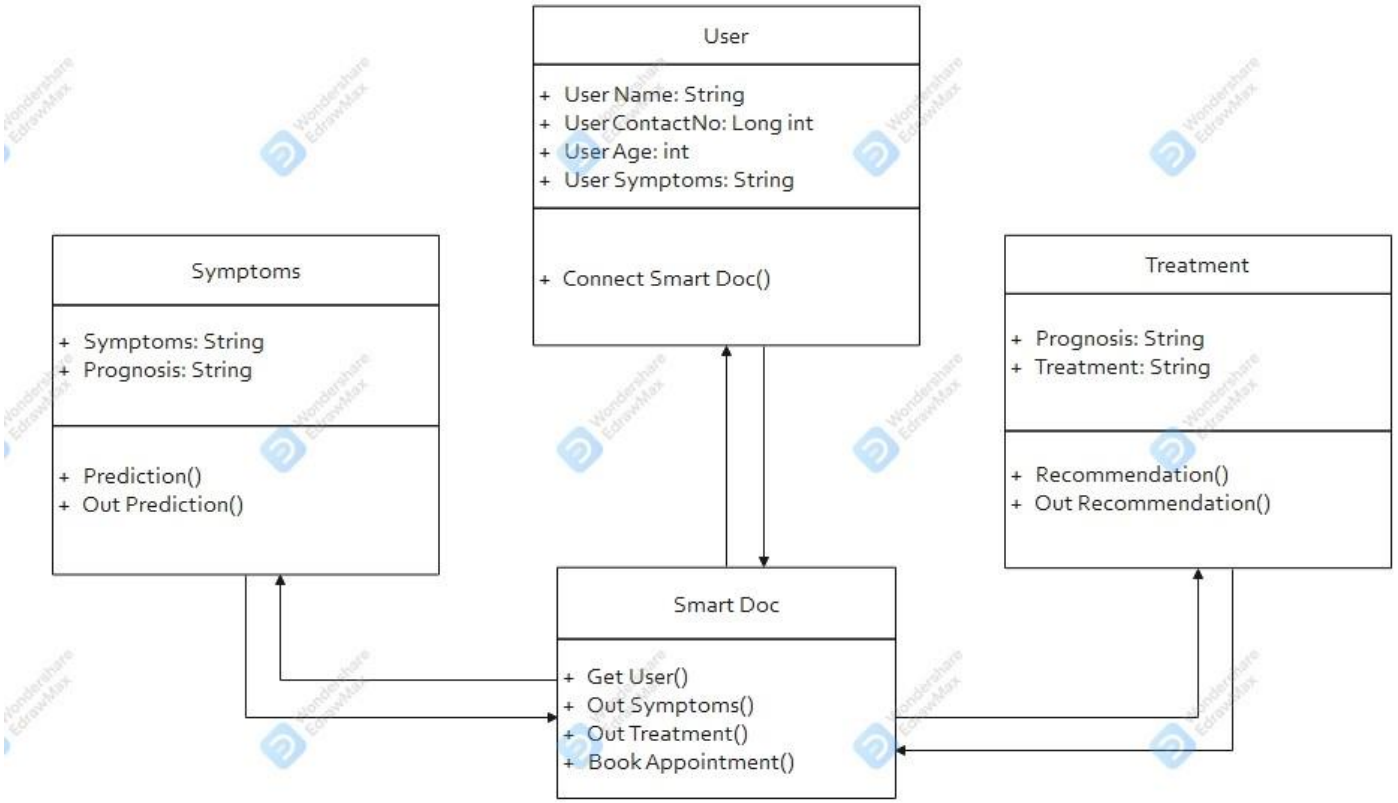
Architecture Diagram



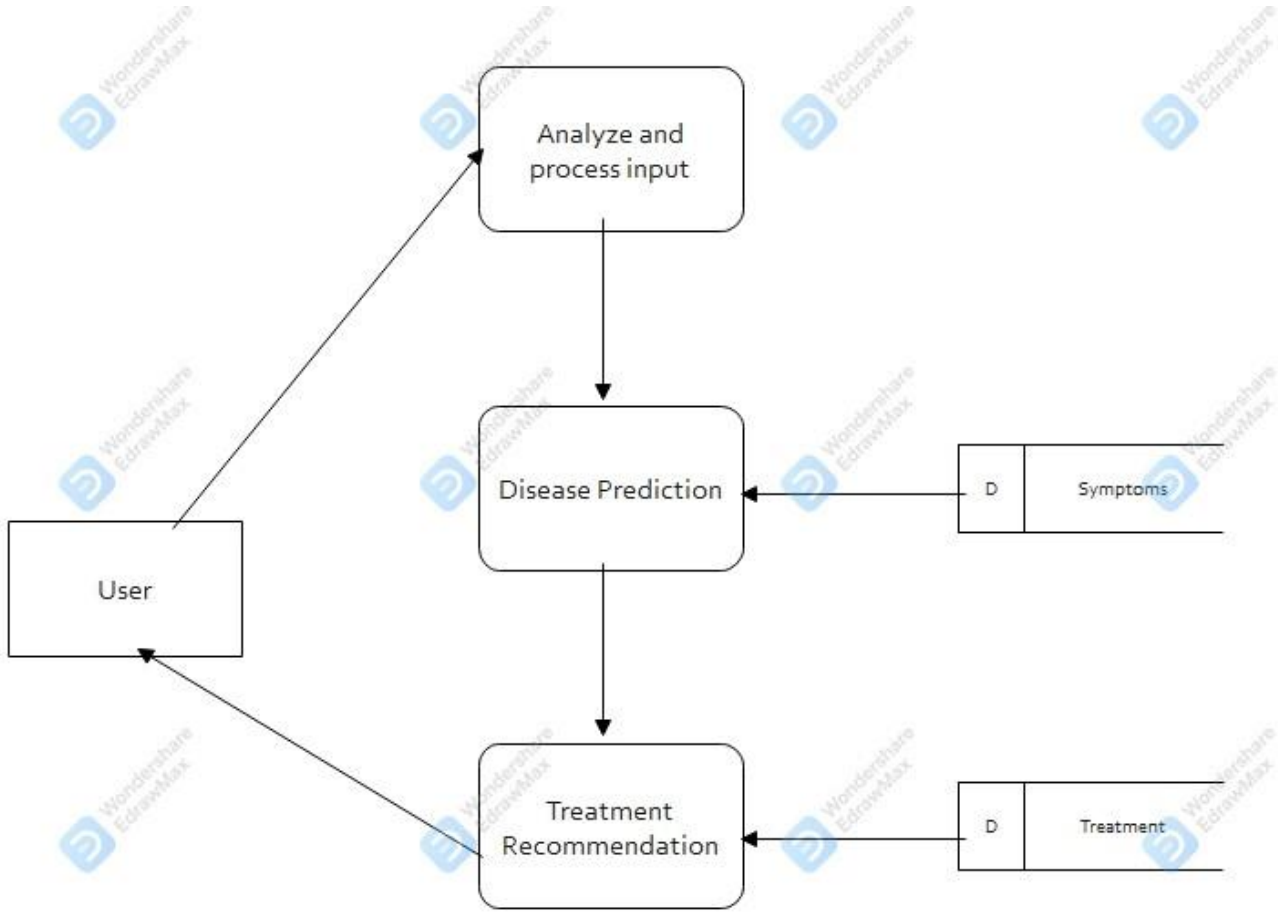
Use Case Diagram



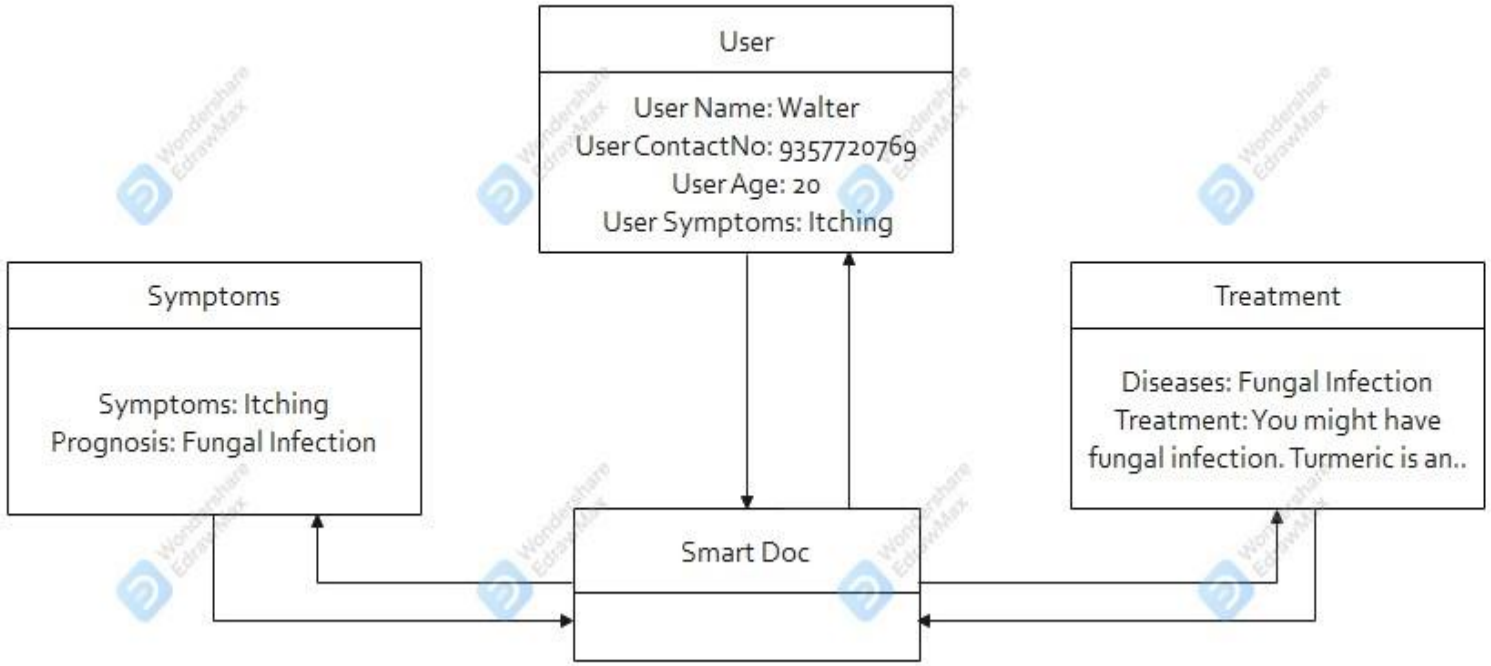
Class Diagram



Data Flow Diagram



Object Diagram



Modules

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.

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.

Modules

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.

Algorithms:

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.

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.

Gaussian Naive 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.

Modules

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 Java script. When a user clicks or performs any action a java script 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.

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.

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

Code

static > css > # style.css > .msg.user

```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));
}


/*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;
```

templates > index.html

```html
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <title>Smart Doc</title>
    <link rel="shortcut icon" type="image/x-icon" href="{{ url_for('static'
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',filena
</head>
<body>
    <script src="https://code.jquery.com/jquery-3.6.0.js" integrity="sha256
    <script type="text/javascript" src="{{ url_for('static',filename='scrip
    <div class="main">
        <!--Mic Action-->
        <div class="microphoneAction" id="microphoneAction">
            <div>Speak</div>
        </div>
        <!--Stop-->
        <div class="stop" id="stop">
        <img src="{{url_for('static',filename='images/stop.png')}}" alt="st
        </div>
        <!--Chat window-->
        <div class="sub" id="sub">
            <div class="msg bot" id="bot">Hello, I am Smart Doc! And your g
            <!--<div class="msg user" id="user">Hello, I am Aishu!</div>-->
        </div>
        <!--Input tag-->
        <div class="chat" id="chat">
            <textarea autocomplete="off" id="send" placeholder=" Type messa
            <img src="{{url_for('static',filename='images/mic.png')}}" alt=
        </div>
    </div>
</body>
</html>
```

Code

static > script > JS script.js > myClick

```
var number = 0;

//Function that replies back
function botAns(userInput) {
    //split the input into an array of strings whenever a blank space is encour
    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
    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) {
```

app.py > ...

```
#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(%
        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)
```

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

Code

response.py > symptoms

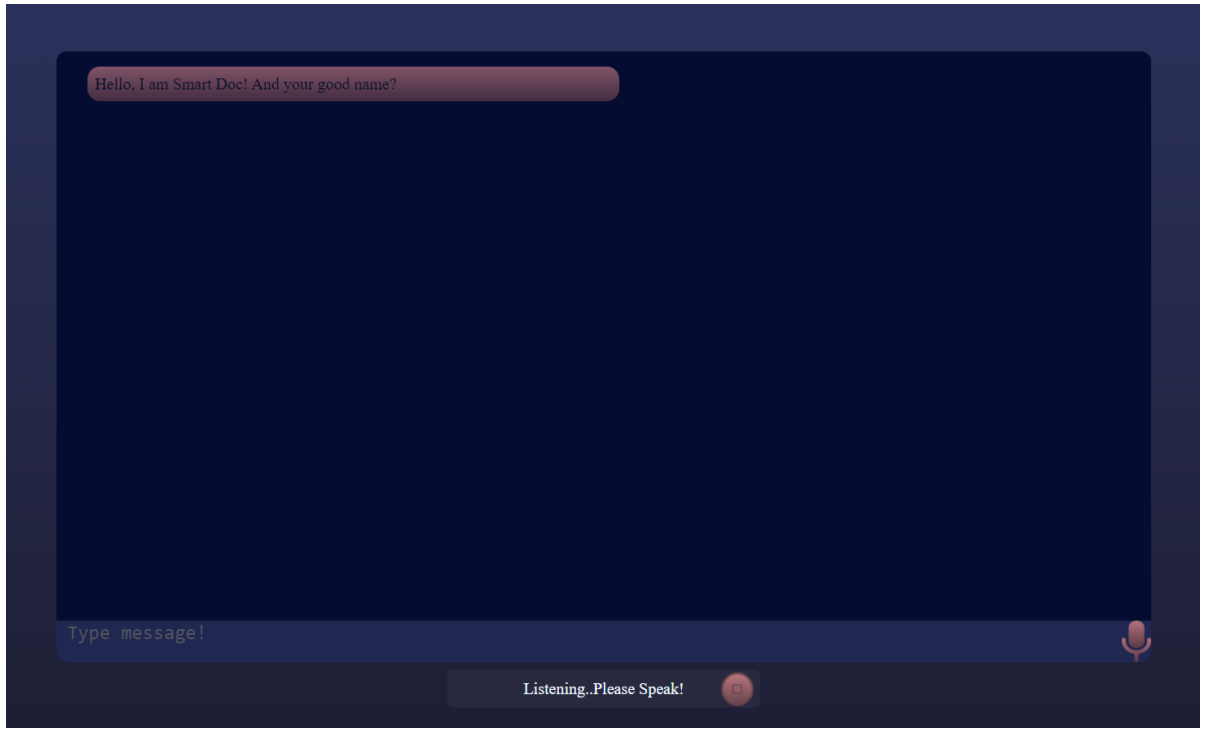
```python
# Function to return the answer based on user input
def check_all_messages(user_input):
    message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
    highest_prob_list = {}


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


    # Responses -------------------------------------------------------
    response(hello(),['hello','hi','hey','hai','smart','doc','hei','hay'
    response(greetings(),['morning','mrng','afternoon','noon','nun','eve
    response(bye(),['bye','goodbye','bubye','tata'], single_response=Tru
    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'], s
    response(thanks(),['thank','thanks'], single_response=True)
    response(yeah(),['yeah','sure','yes','definitely','yup', 'ok','okie'
    response(pain(),['i','have','got','not','feeling','paining','sick','
    response(book_appointment(),['book','appointment', 'doctor','schedul

    best_match = max(highest_prob_list, key=highest_prob_list.get)
    # print(highest_prob_list)
    # print(f'Best match = {best_match} | Score: {highest_prob_list[best
    if highest_prob_list[best_match] < 1:
        return symptoms(user_input)
    else:
       return best_match
```

predict.py > cure

```python
# Defining the Function Input: string containing symptoms separated by commmas Out
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(in
        nb_prediction = data_dict["predictions_classes"][final_nb_model.predict(in
        svm_prediction = data_dict["predictions_classes"][final_svm_model.predict(

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

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

Demo Screenshot



Thank you