HEALTH CHECK CHATBOT

A project report submitted in partial fulfilment of

the requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLICATION

Submitted By

K. DHEERAJ REDDY	(**Regdno-**2018-1902004)
MEGHNA GHALE	(**Regdno**-2018-1902019)
L.VAMSI	(**Regdno**-2018-1902008)

Under the Esteemed Guidance of

Smt. CH. SUNEETHA

Assistant Professor

DEPARTMENT OF COMPUTER APPLICATION

GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND P.G. COURSES (AUTONOMOUS)

(Affiliated to Andhra University)

RUSHIKONDA, VISAKHAPATNAM

2018-2021

GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE and P.G COURSES (Autonomous)

(Affiliated to Andhra University)

RUSHIKONDA, VISAKHAPATNAM

DEPARTMENT OF COMPUTER APPLICATION

CERTIFICATE

This is to certify that the project report entitled "HEALTH CHECK CHATBOT" is the bonafide record of project work carried out by K. DHEERAJ REDDY (Regdno-2018-1902004), MEGHNA GHALE (Regdno-2018-1902019) and L. VAMSI (Regdno-2018-1902008) are students of this college, during the academic year 2018-2021, in partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Application.

Project Guide
Smt. CH. SUNEETHA
Assistant Professor

Head of the Department
Mr. P. Venkata Rao
Associate Professor

External Examiner

DECLARATION

We **K. DHEERAJ REDDY, MEGHNA GHALE and L. VAMSI** hereby declare that the project report entitled "**HEALTH CHECK CHATBOT"** is an original work done at **Gayatri Vidya Parishad College for Degree And P.G Courses (A), Visakhapatnam,** submitted in partial fulfilment of the requirements for the award of Bachelor of Computer Application, to Gayatri Vidya Parishad College for Degree And P.G Courses (A), affiliated to Andhra University. We assure that this project is not submitted in any other University or college.

(K. Dheeraj Reddy)

(Meghna Ghale)

(L. Vamsi)

ACKNOWLEDGEMENT

I consider it as a privilege to thank all those people who helped me a lot for successful completion of the project "**HEALTH CHECK CHATBOT**"

First of all, I would like to thank our beloved **principal Prof. S. Rajani** of **Gayatri Vidya Parishad College for Degree And P.G Courses(A),** who has provided full- fledged lab and infrastructure for successful completion of my project work.

I would like to thank our ever-accommodating Head of the Department, Bachelor of Computer Applications **Sri P. VENKATA RAO, Associate Professor** and our guide **Smt. CH. SUNEETHA, Assistant Professor** has obliged in responding to every request though they are busy with their hectic schedule of administration and teaching.

I thank all the **Teaching & Non-Teaching staff** who has been a constant source of support and encouragement during the study tenure.

(K. Dheeraj Reddy)

(Meghna Ghale)

(L. Vamsi)

ABSTRACT

ABSTRACT

CHATBOT USING DEEP LEARNING ALGORITHMS

In our lives, health is extremely vital. It is quite difficult to seek a doctor's consultation for a basic health condition in this covid epidemic circumstance. This Project aims on developing a medical chatbot using Artificial Neural Network concept which is programmed in a way to diagnose the disease and provide a general prescription and basic remedies which a person can relay to some extent. More or less, it helps in providing the information as programmed to respond within a short time and ease in accessibility for the user. To do so, thorough research is performed and programmed by analysing the probability of the text input.

HEALTH CHECK CHATBOT

INDEX

INTRODUCTION

INTRODUCTION

CHATBOT

Chat bots are potentially referred to as human-machine interactions. Eventually, these virtual agents are getting involved in the main global sectors such as healthcare, banking, education, agriculture, etc. It's a kind of artificial intelligence (AI) interaction between the users and machines with the intervention of deep learning classification algorithm.

Because the healthcare industry is so strongly linked to human interaction, it seems contradictory that conversational AI technologies such as chat bots are becoming more popular. Hospital administrators are spending their days in appointing schedules and answering routine questions for the patients which is time consuming, such jobs can be easily fixed using a bot application.

A chatbot is a computer program designed to simulate an intelligent conversation with human users in natural language via voice or textual methods.

Chatbots are versatile enough to carry out conversations and even tasks for the users. They can be as simple as programs that answer a simple query with a single-line response, or as complex as digital assistants that learn and evolve to deliver increasing levels of personalization.

BOT: A chatbot is a service powered by rules and artificial intelligence.

USER: Users communicate with a chatbot via the chat interface, similar to how they would talk to a real person.

History Of Chatbots:

The first chatbot ever was developed by MIT professor Joseph Weinbaum in the 1960s. It was called ELIZA. It was in 1994 that the term 'Chatterbot" was coined. You'll read more about ELIZA and other popular chatbots that were developed in the second half of the 20th century later on. In the year 2009, a company called WeChat in China created a more advanced Chatbot.

How do Chatbots work?

Chatbots or digital assistants use Natural Language Processing to communicate with humans. Known as **NLP**, this technology focuses on understanding how humans communicate with each other and how we can get a computer to understand and replicate this behaviour.

For example, if we say “Hello” to a chatbot, it understands that this is similar to ‘Hi’ or ‘Good morning’ because we taught the machine to understand so. This is a simple example, but sometimes the same word can mean different things. There is a big difference between “I’m fine” and “Pay the fine”. Just like humans, computers need context to learn what you mean. Chatbots are trained to understand the difference with the use of machine learning. One of the gigantic industries that have benefited from the advent of chatbots is Healthcare.

Monitoring health: Chatbots can act as an authentic assistant by reminding them to sip water, pop a tablet, or provide healthy tips.

Suggesting doctors: Doctors to get overwhelmed with the enormous number of names in the medical field. Chatbots can quickly help them with a medicine name, suggest a doctor or dosage, or standards.

Advantages of chatbots in the healthcare industry:

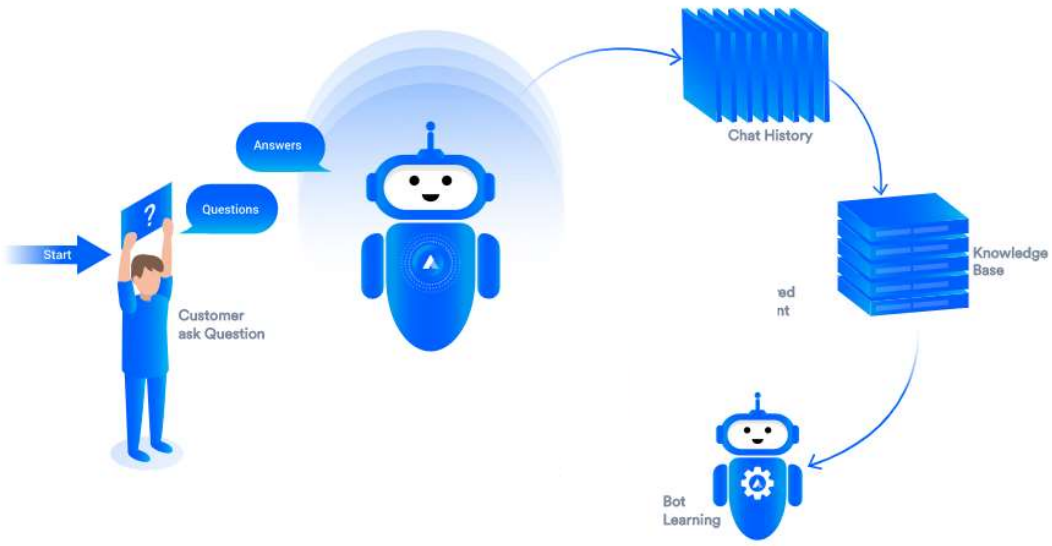
- Availability and ongoing health monitoring. All healthcare providers are always willing to help their patients.
- Providing information fast when there is not a moment to lose. Emergencies are solved quicky.
- Gaining the trust of patients. All medical institutions keeping up with advancing technology should be effectively used.
- Scheduling appointments. As we just have read in the previous item, virtual assistants for healthcare websites.

LITERATURE SURVEY

LITERATURE SURVEY

2.1 INTRODUCTION

Chatbots are potentially referred to as human-machine interactions. Eventually, these virtual agents are getting involved in the main global sectors such as healthcare, banking, education, agriculture, etc. It's a kind of artificial intelligence (AI) interaction between the users and machines with the intervention of deep learning classification algorithm. Because the healthcare industry is so strongly linked to human interaction, it seems contradictory that conversational AI technologies such as chatbots are becoming more popular. Hospital administrators are spending their days in appointing schedules and answering routine questions for the patients which is time consuming, such jobs can be easily fixed using a bot application.

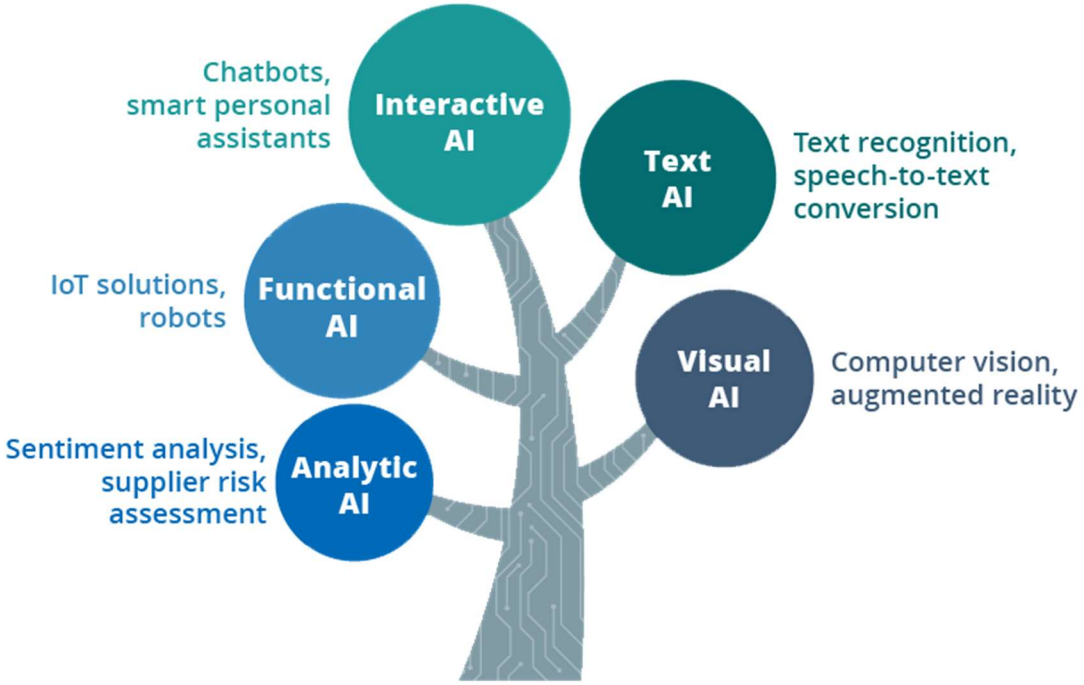


2.2 CURRENT SYSTEM

In our lives, health is extremely vital. It is quite difficult to seek a doctor's consultation for a basic health condition in this covid epidemic circumstance. This Project aims on developing a medical chatbot using Artificial Neural Network concept which is programmed in a way to diagnose the disease and provide a general prescription and basic remedies which a person can relay to some extent. More or less, it helps in providing the information as programmed to

respond within a short time and ease in accessibility for the user. To do so, thorough research is performed and programmed by analysing the probability of the text input.

2.3 CHATBOT CLASSIFICATION



Analytic AI

Powered with machine learning (including its most advanced deep learning techniques), analytic AI scans tons of data for dependencies and patterns to ultimately produce recommendations or provide a business with insights, thus contributing to data-driven decision-making.

Sentiment analysis and supplier risk assessment are just a few examples of analytic AI in action. If you'd like to get a complete picture of how such a solution works, our experts have summarized the insights gained from their experience with two of the use cases – inventory optimization and demand forecasting.

Functional AI

Functional AI is very similar to analytic AI – it also scans huge amounts of data and searches for patterns and dependencies in it. However, instead of giving recommendations, functional AI takes actions. For instance, being the part of the IoT cloud, it can spot a machine-breakdown pattern in the sensor data received from a certain machine, and trigger a command to turn this machine off. Another example: robots that Amazon uses to bring the shelves with the goods to the pickers, thus speeding up the picking process.

Interactive AI

This type of AI allows businesses to automate communication without compromising on interactivity. To envisage this type of AI, think of chatbots and smart personal assistants whose abilities can vary from answering pre-built questions to understanding the conversation context.

Interactive AI can serve another purpose – improving a company's internal processes. For example, one of our projects was dedicated to creating a chatbot to facilitate the corporate process of vacation booking.

Text AI

Businesses that use text AI can enjoy text recognition, speech-to-text conversion, machine translation, and content generation capabilities. Even if a company is not Google or Amazon, or any other giant company that provides text AI as a service, it can still take advantage of this AI type. For example, the company can use text AI to power an internal corporate knowledge base.

Visual AI

With visual AI, businesses can identify, recognize, classify and sort objects or convert images and videos into insights. A computer system that helps an insurer to estimate damage based on damaged car photos or a machine that grades apples based on their colour and size are the examples of visual AI. This type of AI covers computer vision or augmented reality fields.

2.4 PROPOSED SYSTEM

- Indian developed chatbots are still under development in healthcare field and usage of these bots are limited due to a lack of sources. Some of the widely used general chatbots are unable to diagnose health problems as they depend on internet search.
- This project helps the people with covid symptom diagnosis, common medications, and general prescriptions through ease in accessibility. This chatbot responds to the user's command's using NLP (Natural Language Processing), NLTK (Natural Language Toolkit) and classification algorithms i.e., Artificial Neural Network (ANN) with an activation function.

2.5 OBJECTIVES

The objectives of the project are as follows:

Non-technical objectives:

- Cost Savings
- 24/7 Availability
- Instant Response
- Multilingual
- Omni-Channel
- Bots Save a Great Deal of Time
- Consistency In Answers
- Seamless Transactions
- Applicable To Multiple Industries

Technical objectives:

- Search of required dataset.
- Loading the dataset.
- Preprocessing of dataset.
- Standardization or Normalization of dataset.
- Model creation.
- Model training and testing.
- Text & Disease Prediction.
- Model accuracy report

2.6 REQUIREMENTS ELICITATION

Requirement is the feature the system must have a constraint that it must satisfy to be accepted by the clients. Requirement's engineering aims at defining the requirements of the system under construction. It includes two main activities namely Requirements Elicitation and Analysis.

Requirements Elicitation focuses on describing the process of the system. The client, the developer and the users identify the problem. This specification is structured and formulized during analysis to produce an Analysis Model. Requirements Elicitation and Analysis focuses only on the user's view of the system.

Requirements Elicitation includes the following activities:

Identifying Actors:
During this activity, developers identify the different types of users the future system will support.

Identifying Scenarios:
During this activity, developers observe users and develop a set if detailed scenarios for typical functionality provided by the future system. Developers use these scenarios to communicate with the users and deepen their understanding.

Identifying Use Cases:
Once developers and users agree on the set of scenarios, a set of use cases that completely represent the future system.

Refining Use Cases:
During this activity, developers ensure that the requirements specifications completed by detailing each use case and describing the behavior of the system in the presence of errors and exceptional conditions.

Identifying Relationships Among Use Cases:
During this activity, developers identify the dependencies among the use cases and also consolidate the use case model by factoring out common functionality.

Identifying Non-functional Requirements:
During this activity, developers, users and clients agree on aspects like performance of system, documentation, resources, security and its quality.

- **Actors:**

Actors are external entities that interact with the system. They have a unique names and descriptions. In this project the two actors are required user (patient)and bot.

2.6.1 FUNCTIONAL REQUIREMENTS

It describes the interactions between the system and its environment independent of its implementation.

The functional requirements are:

- Intents Dataset
- Dengue Dataset
- Python modules
    - Pandas
    - Nltk
    - Tensorflow
    - Tflearn
    - Keras
    - Random
    - Json
    - Web browser
    - Tkinter
    - Numpy
    - Matplotlib
    - Sklearn
    - PIL
- Spyder IDE

INTENTS:

Within a chatbot, intent refers to the goal the customer has in mind when typing in a question or comment. While entity refers to the modifier the customer uses to describe their issue, intent is what they really mean.

DATA EXTRACTION:

Data Extraction is the process of obtaining data for further processing the bot extracts data from the user for further data pre-processing.

DATA PREPROCESSING:

In data pre-processing, stemming and Bag of words are used

- **STEMMING:**

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”. Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words. How do we get these tokenized words? Well, tokenization involves breaking down the document into different words.

- **BAG OF WORDS:**

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Bag of Words model is used to pre-process the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words. This model can be visualized using a table, which contains the count of words corresponding to the word itself.

NLTK:

Natural language processing (NLP) is a field that focuses on making natural human language usable by computer programs. NLTK, or Natural Language Toolkit, is a Python package that you can use for NLP. A lot of the data that you could be analysing is unstructured data and contains human-readable text

MODEL CREATION:

Deep learning models are built using neural networks. A neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training. Then the model spits out a prediction. The weights are adjusted to find patterns in order to make better predictions.

DNN:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions.

ANN:

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

Neurons: -

Within an artificial neural network, a neuron is a mathematical function that model the functioning of a biological neuron. Typically, a neuron computes the weighted average of its input, and this sum is passed through a nonlinear function, often called activation function, such as the sigmoid.

Weights & biases:

Weights and biases (commonly referred to as w and b) are the learnable parameters of some machine learning models, including neural networks. Weights control the signal two neurons. In other words, a weight decides how much influence the input will have on the output.

- Model1: - Feedforward neural network (FFNN)

Feedforward neural networks are also known as Multi-layered Network of Neurons (MLN). These networks of models are called feedforward because the information only travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes.

- Model2: - Sequential model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

MODEL TRAINING:

Model training is the phase where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range.

OUTPUT:

The reply given by the chatbot is considered as output.

Output is given by picking up the highest probability case.

2.6.2 NON-FUNCTIONAL REQUIREMENTS

During this activity, developers, users and clients (sender and receiver) agree on aspects like performance of system, documentation, resources and its quality.

The following are the non- functional requirements of the system:

- **Usability:**

  The interface of this system is GUI. The user interface could be user friendly. So that the users can easily use the system and the user interface could be flexible.

- **Reliability:**

  The system is designed in such a way that it produces correct results.

- **Correctness:**

  This System functions well in the presence of invalid inputs and displays error Messages to users.

- **Performance:**

  It displays the correct result based on the input provided in fast and robust day.

- **Scalability**:

  The highest workloads under which the system will still perform as expected.

- **Portability:**

  Software should run on any operating systems, browsers, and their versions.

- **Compatibility:**

  Software should compatible with other application and processes within these environments.

- **Security:**

  System should be secure and robust.

System Requirements:

- **OS**: Windows 7 with SP1;
- **Recommended**: Windows 10.
- **CPU**: Intel or AMD processor with 64-bit support;
- **Recommended**: 2.8 GHz or faster processor.
- **Disk Storage**: 2 GB of free disk space.
- **Monitor Resolution**: 1280x800;
- **Recommended**: 1920x1080.
- **Internet**: Internet connection required for web search.

2.7 SOFTWARE AND TECHNOLOGIES DESCRIPTION

History

The programming language Python was conceived in the late 1980s and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to ABC capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, Benevolent Dictator for Life (BDFL). (However, van Rossum stepped down as leader on July 12, 2018.). Python was named after the BBC TV show Monty Python's Flying Circus.

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

IDE (Integrated Development Environment):

The IDE is an extraordinary text editor that makes the overall development job easier for anyone. There are a plenty of editing functions that are specific to the language. This is very fast and comfortable in use. The Python IDE also work as an interpreter sometimes.

Python provides a number of integrated development environments (IDE) that is supports by the windows operating system.

They are:

- sublime text 3
- PyCharm
- eclipse with pydev
- spyder wing
- Eric

Fig 2.4

Python Features

- **Integrated**
- Easy to Learn and Use
- **Expressive language**
- **Interpreted Language**
- **Cross-Platform Language**
- **Free and Open Source**
- **Object-Oriented Language**
- **Extensible**
- **Large Standard Library**
- GUI Programming Support
- Dynamic Memory Allocation
- **Wide-Ranging Support libraries**

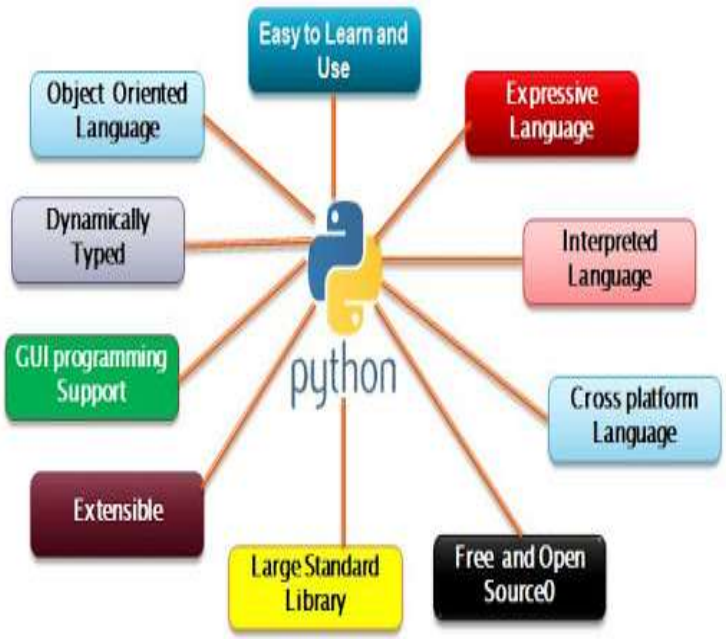


Fig 2.5

Tkinter Module

Tkinter is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest.

Developing desktop-based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. Import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Tkinter Widgets

There are various widgets like button, canvas, check button, entry, etc. that are used to build the python GUI applications.

- Button- The Button is used to add various kinds of buttons to the python application.

- Canvas- The canvas widget is used to draw the canvas on the window.
- Checkbutton- The Checkbutton is used to display the Check button on the window.
- Entry- The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
- Frame- It can be defined as a container to which, another widget can be added and organized.
- Label- A label is a text used to display some message or information about the other widgets.
- ListBox- The ListBox widget is used to display a list of options to the user.
- MenuButton- The MenuButton is used to display the menu items to the user.
- Menu- It is used to add menu items to the user.
- Message- The Message widget is used to display the message-box to the user.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

- The pack () method

  The pack () widget is used to organize widget in the block. The positions widgets added to the python application using the pack () method can be controlled by using the various options specified in the method call. Syntax- widget. Pack(options)

- The grid () method

  The grid () geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget. Syntax- widget.grid(options)

- The place() method

  The place() geometry manager organizes the widgets to the specific x and y coordinates. Syntax- widget.place(options)

UML MODELING

UML MODELING

3.1 INTRODUCTION

UML:

The Unified Modeling Language (UML) is a well-recognized graphical modeling language for Object-Oriented software engineering. UML has been widely used by software engineers for the specification, construction and documentation of software systems. UML provides a systematic way of describing a system's architecture, capturing its static, functional and dynamic views in detail. Using UML, system analysts, designers and programmers can build and maintain a complex software system much more easily and, at the same time, ensure that the system will hold up to requirement change.

One of the purposes of UML was to provide the development community with a stable and common design language that could be used to develop and build computer applications. UML years. Using UML, IT professionals could now read and disseminate system structure and design plans -- just as construction workers have been doing for years with blueprints of buildings.

UML Standard Diagrams:

The elements are like components which can be associated in different ways to make complete UML pictures which is known as diagram. So, it is very important to understand the different diagrams to implement the knowledge in real life systems.

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. So, if we look around then we will realize that the diagrams are not a new concept but it is used widely in different form in different industries.

We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So, UML defines various kinds of diagrams to cover most of the aspects of a system.

You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way.

There are two broad categories of diagrams and then are again divided into sub-categories:

- Structural Diagrams
- Behavioural Diagrams

Structural Diagrams:

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

These static parts are represented by classes, interfaces, objects, components and nodes. The four structural diagrams are:

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioural Diagram:

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are covered fully.

Behavioural diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioural diagrams:

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

3.2 INTRODUCTION TO USE CASE DIAGRAM

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicts the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Use Cases:

Use cases describe the behaviour of the system as seen from the actor's point of view. It describes the function provided by the system as a set of events that yield a visible result for the actors.

Actors initiate the use cases for accessing system's functionality. When actors and use cases exchange information, they are said to **Communicate.** To describe a use case, we use a template composed of six fields:

Use Case Name:	The name of the use case
Participating Actors:	The actors participating in the particular use case
Entry Condition:	Condition for initiating the use case.
Flow of events:	Sequence of steps describing the functioning of use cases.
Exit Condition:	Condition for terminating the use case.

Quality Requirements:

Requirements that do not belong to the use case but Constraint the functionality of the system.

Use case diagrams include four types of relations. They are as follows:

- Communication Relationships
- Inclusion Relationships
- Extension Relationships
- Inheritance Relationships

ACTORS

Actor:

- Actors are external entities that interact with the system. Actors typically include a user role or another system. They have unique names and descriptions

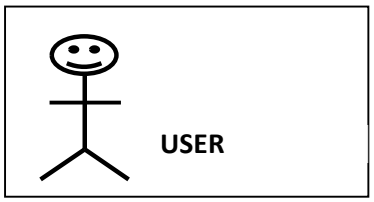


Fig 3.1.1

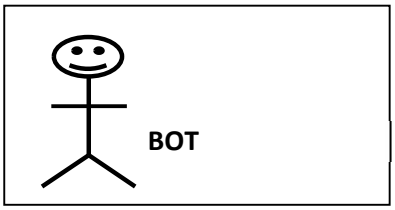


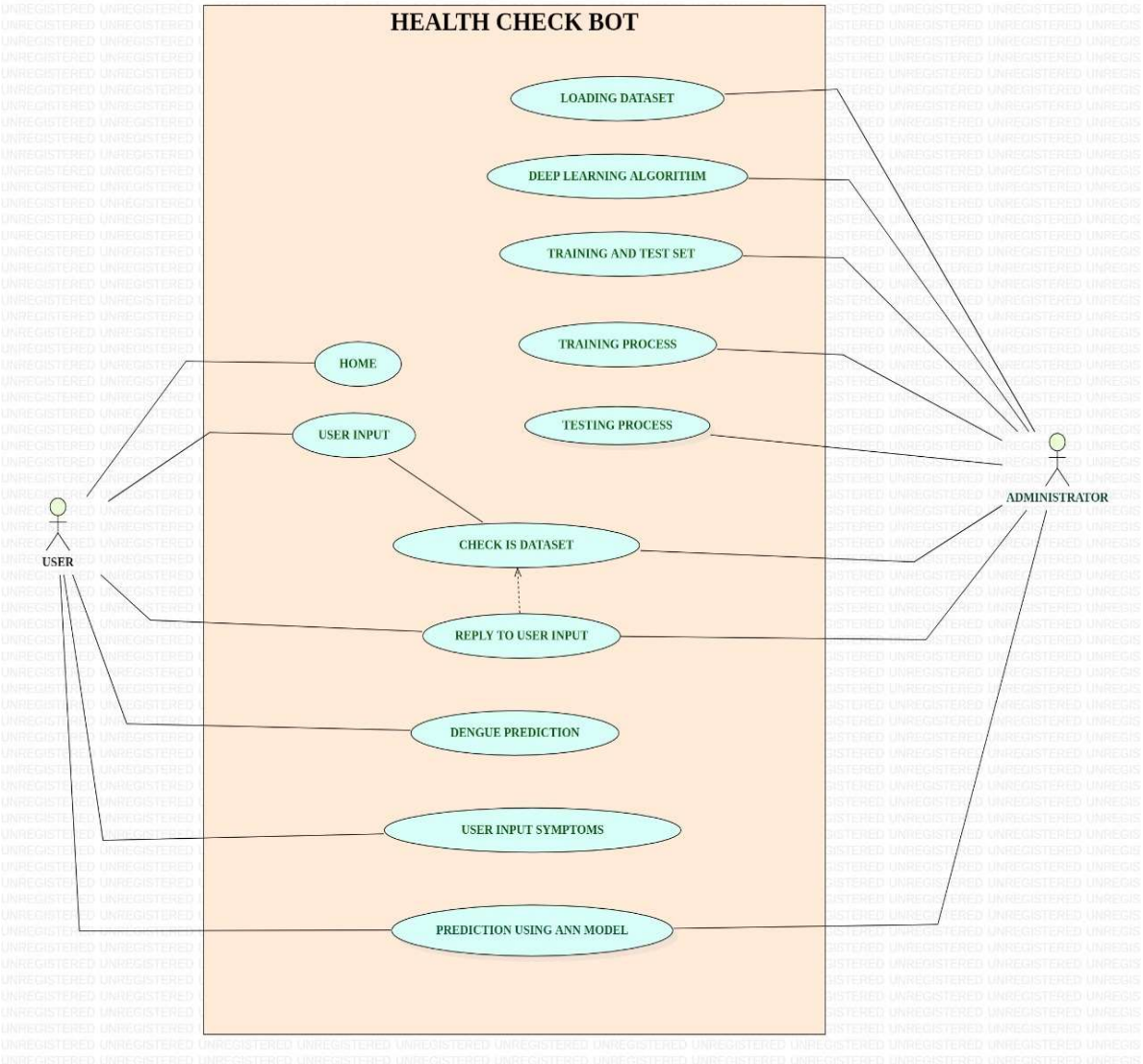
Fig 3.1.2

3.2.1 USE CASE DIAGRAM

Description:

In the use case diagram, the user enters the query and the bot classifies the intents and generates the best response for the user asked query. If the query matches to the bot database it shows the output otherwise it returns back to the main menu (chat)

USE CASE DIAGRAM FOR CHATBOT



1.User Use case:

Use Case Name	Health check chatbot
Participating Actor	patient
Entry Condition	Patient enters query
Flow of Events	1. search for intents 2. search suitable answer
Exit Condition	Exit command

Table 1: Users Use case

2.Bot Use case:

Use Case Name	Health check chatbot
Participating Actors	Sender: Bot
Entry Condition	Training & Testing
Flow of Events	1. search for intents 2. search suitable answer
Exit condition	Output

Table 2: Bot Use case

Scenario:

A Scenario is an instance of a use case describing a concrete set of actions. Scenarios are used as examples for illustrating common cases: their focus is on understandability. To describe a scenario, we use a template composed of three fields:

Scenario Name : The name of the Scenario

Participating Actors : The Instances of the actors participating in the Scenario.

Flow of events : Sequence of steps describing the events in Scenario.

Scenario 1:

Scenario Name	Output
Participating Actors	Bot
Flow of Events	1. Searches intents 2. Pre-processes data 3. Output

Table 4: Scenario Bot operation

Scenario 2:

Scenario Name	Input
Participating Actors	User
Flow of Events	1. Enters query 2. Get results.

Table 5: Scenario Users operation

3.3 INTRODUCTION TO SEQUENCE DIAGRAM

Interaction diagrams model the behaviour of use cases by describing the way groups of objects interact to complete the task. The two kinds of interaction diagrams are Sequence and Collaboration diagrams. Sequence diagrams generally show the sequence of events that occur. Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass. The diagrams are read left to right and descending. Following are the Sequence Diagrams for the system under consideration.

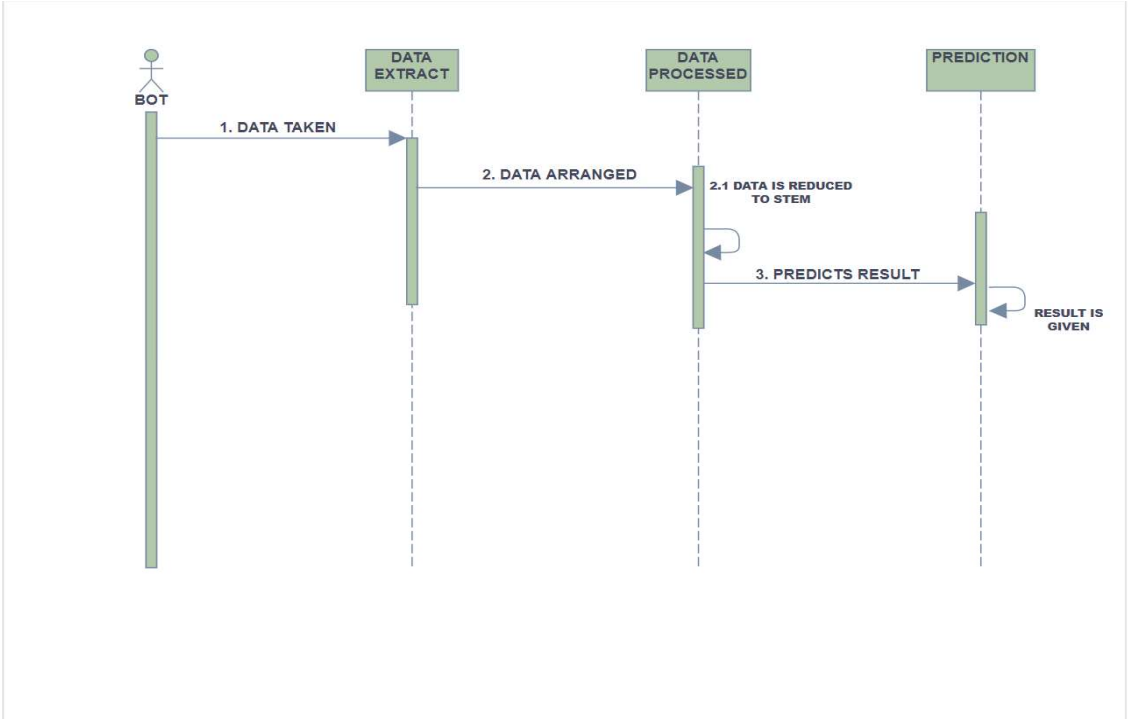
In object-oriented analysis, developers build a model describing the application domain. The analysis model is then extended to describe how the actors and the system together with non-functional requirements, to prepare the architecture of the system developed during high-level design, which is correct, complete, consistent and verifiable.

Analysis object model is represented by class and object diagrams. Analysis focuses on producing a model of the system, called the analysis model, which is correct, complete, consistent, and verifiable. Analysis is different from requirements elicitation in that developer's focus on structuring and formalizing the requirements elicited from users. This formalization leads to new insights and discovery of errors in the requirements. As the analysis model may not be understandable to the users and the client, developers need to update the requirements specification to reflect insights gained during analysis, and then review the changes with the client and users. In the end, the requirements, however large, should be understandable by the client and the users.

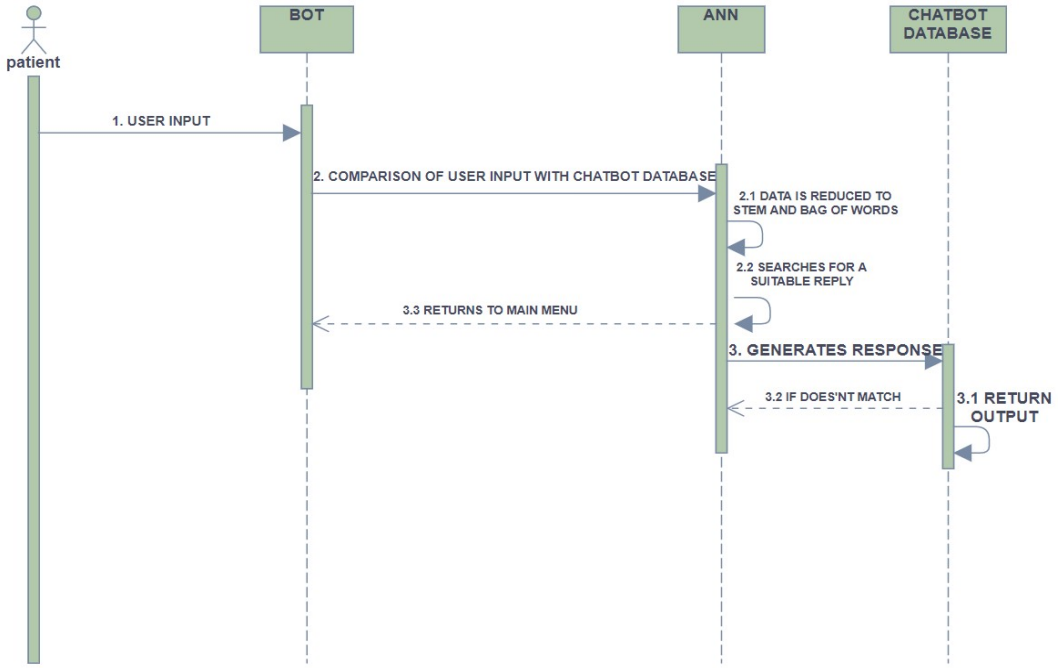
The analysis model is composed of three individual models: the functional model represented by use cases and scenarios, the analysis, represented by class and object diagrams, and the dynamic model, represented by state chart and sequence diagrams. We described how to elicit requirements from the users and describe them as use cases and scenarios. We describe how to refine the functional model and derive the object and the dynamic model. This leads to a more precise and complete specification as details is added to the analysis model. We conclude by describing management activities related to analysis.

3.3.1 SEQUENCE DIAGRAM

Sequence Diagram for Bot



Sequence Diagram for User



3.4 INTRODUCTION TO STATE-CHART DIAGRAM

State diagrams are used to describe the behaviour of a system. State diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system. Not all classes will require a state diagram and state diagrams are not useful for describing the collaboration of all objects in a use case. State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state. All state diagrams being with an initial state of the object. This is the state of the object when it is created. After the initial state the object begins changing states.

Control flow is the sequencing of actions in a system. It defines the order of execution of operations. These decisions are based on external events generated by an actor or on the passage of time.

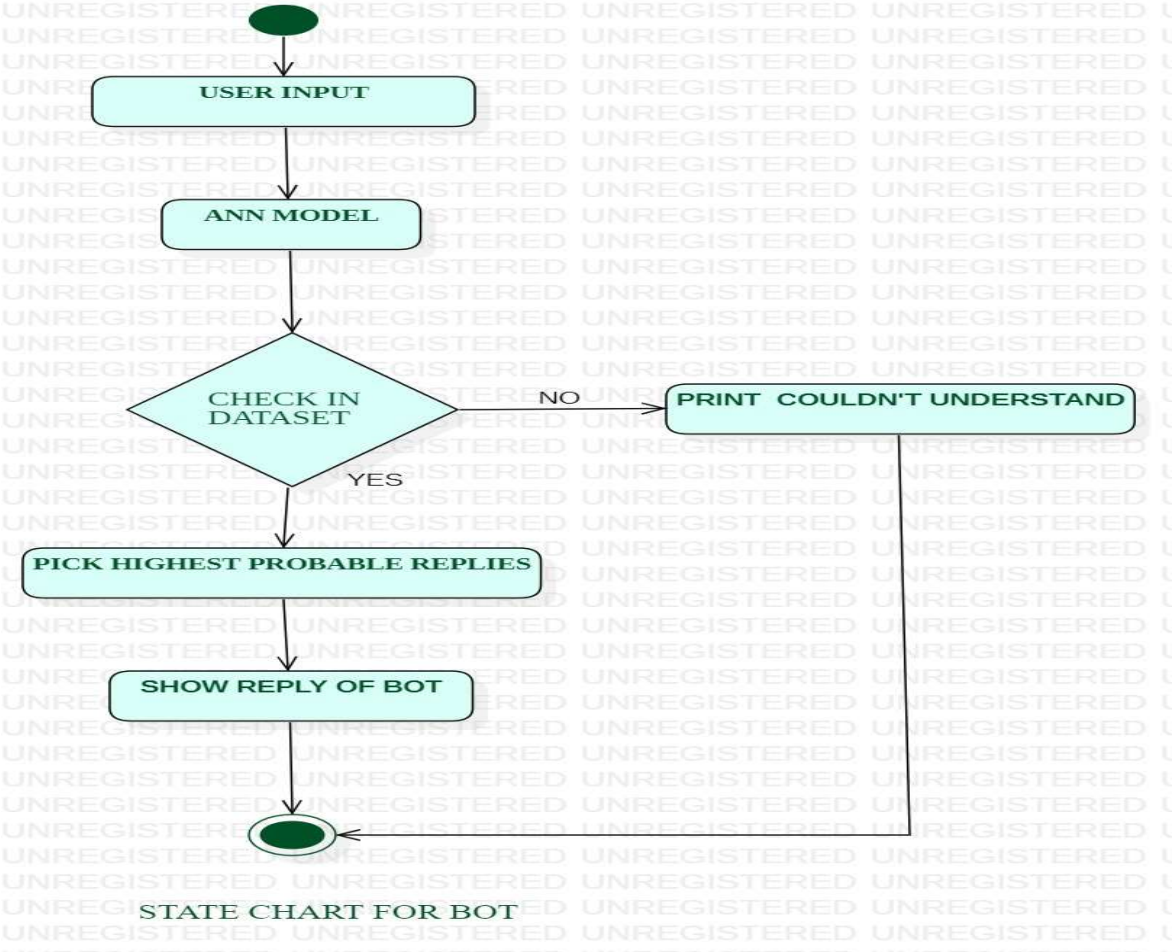
Activity diagrams illustrate the dynamic nature of a system modelling the flow of control from activity to activity. In many ways Uml activity diagrams are the object-oriented

equivalent of flow charts and data-flow diagrams (DFD's) from structured development. They are used to explore the logic of:

- A complex operation
- A complex business rule
- A single use case
- Several use cases
- A business process
- Software processes

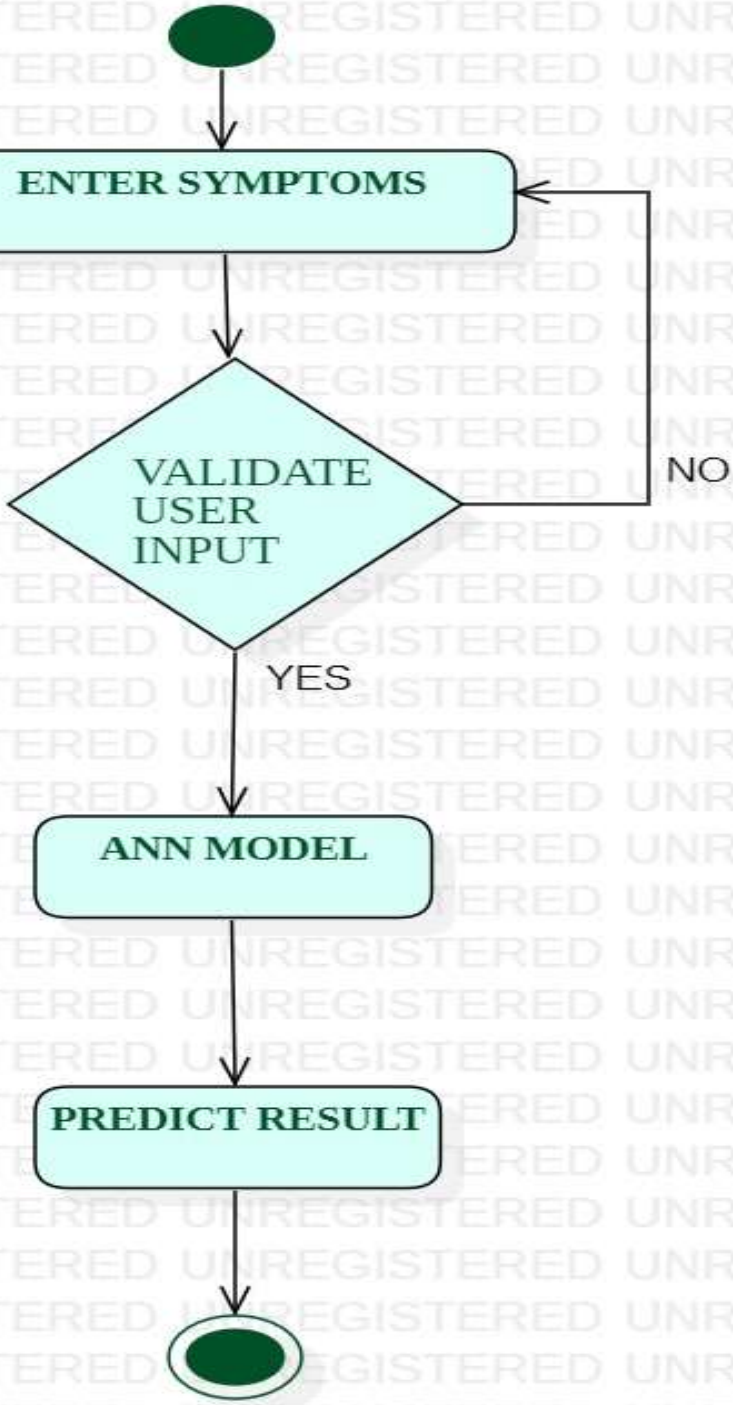
3.4.1 STATE-CHART DIAGRAM

STATE-CHAT FOR BOT



STATE CHART FOR BOT

STATE-CHAT FOR DENGUE



STATE CHART FOR DENGUE PREDICTION

DESIGN

DESIGN

4.1 DESIGN AND DESCRIPTION OF ALGORITHM

Chatbot architecture is the heart of chatbot development. Based on the usability and context of business operations the architecture involved in building a chatbot changes dramatically. So, based on client requirements we need to alter different elements; but the basic communication flow remains the same. Learn how to choose the right chatbot architecture and various aspects of the Conversational Chatbot.

Algorithm:

Step 1: Start

Step 2: Take the users data

Step 3: Use the users data obtained from step (2) and do the following:

- Intents
- Data extraction
- Convert the data into bag using stemming and bag of words
- Then model is created and trained

Step 4: The obtained output is sent to the patient

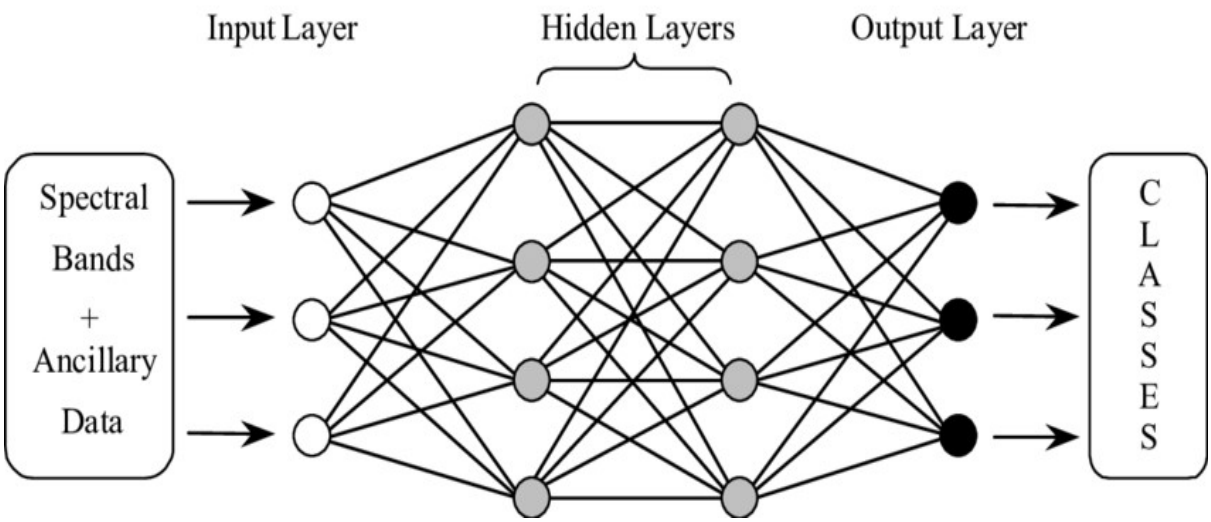
Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

ANN Algorithms used in this project:

FFNN

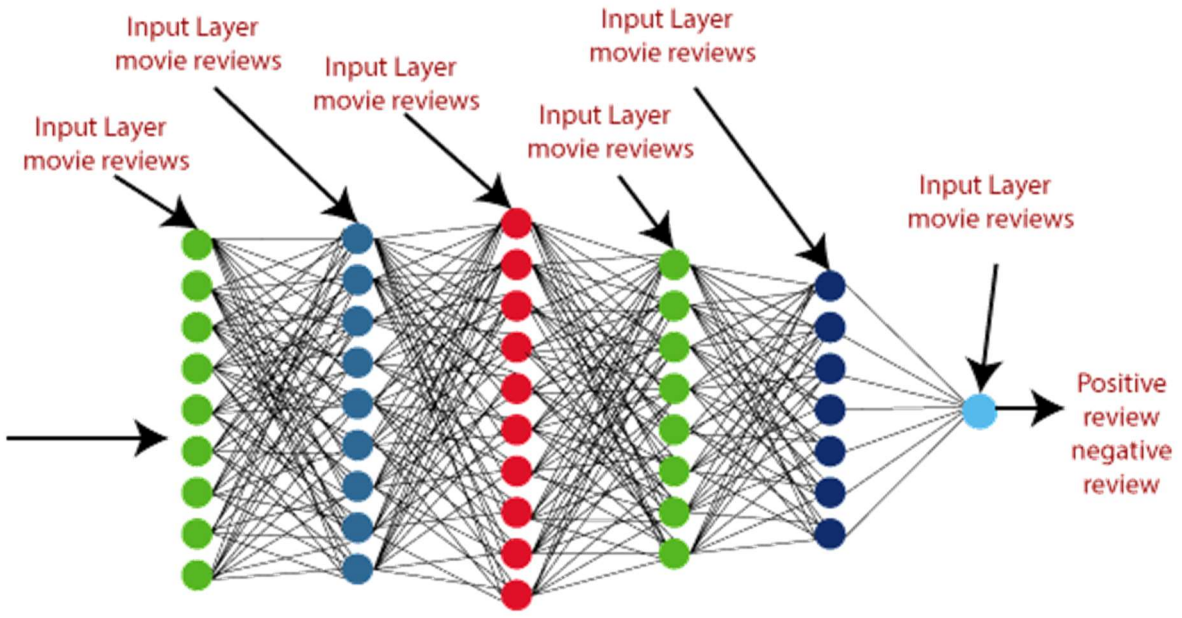
Feedforward neural networks are also known as Multi-layered Network of Neurons (MLN). These networks of models are called feedforward because the information only travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes.



SEQUENTIAL MODEL

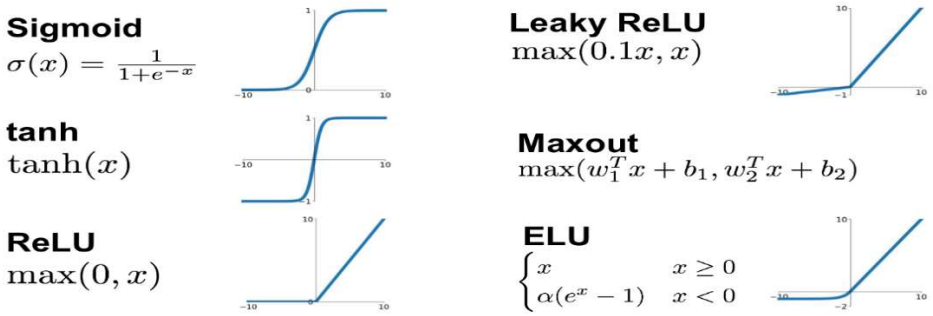
Sequential model is basically a linear composition of Keras Layers. Sequential model is easy, minimal as well as has the ability to represent nearly all available neural networks.

Sequential model exposes Model class to create customized models as well. We can use sub-classing concept to create our own complex model.

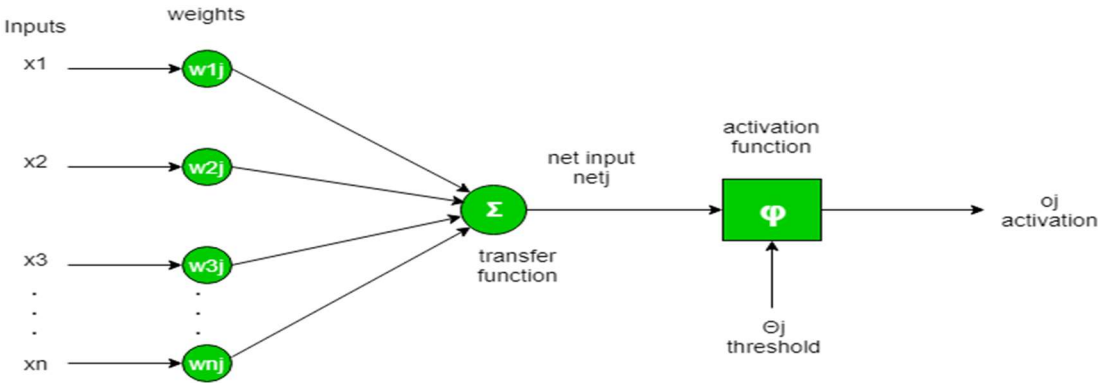


Keras Neural Network Sequential Model

ACTIVATION FUNCTIONS



To put in simple terms, an artificial neuron calculates the 'weighted sum' of its inputs and adds a bias, as shown in the figure below by the net input.

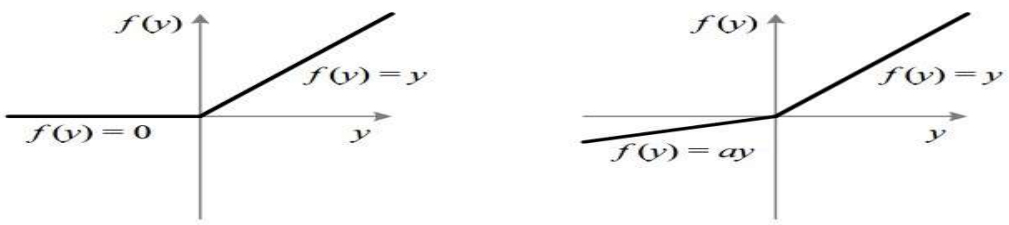


Now the value of net input can be any anything from -inf to +inf. The neuron doesn't really know how to bound to value and thus is not able to decide the firing pattern. Thus, the activation function is an important part of an artificial neural network. They basically decide whether a neuron should be activated or not. Thus, it bounds the value of the net input. The activation function is a non-linear transformation that we do over the input before sending it to the next layer of neurons or finalizing it as output.

ACTIVATION FUNCTIONS USED

ReLU (Rectified Linear Unit)

The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.



The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. If you look at the ReLU function if the input is negative, it will convert it to zero and the neuron does not get activated.

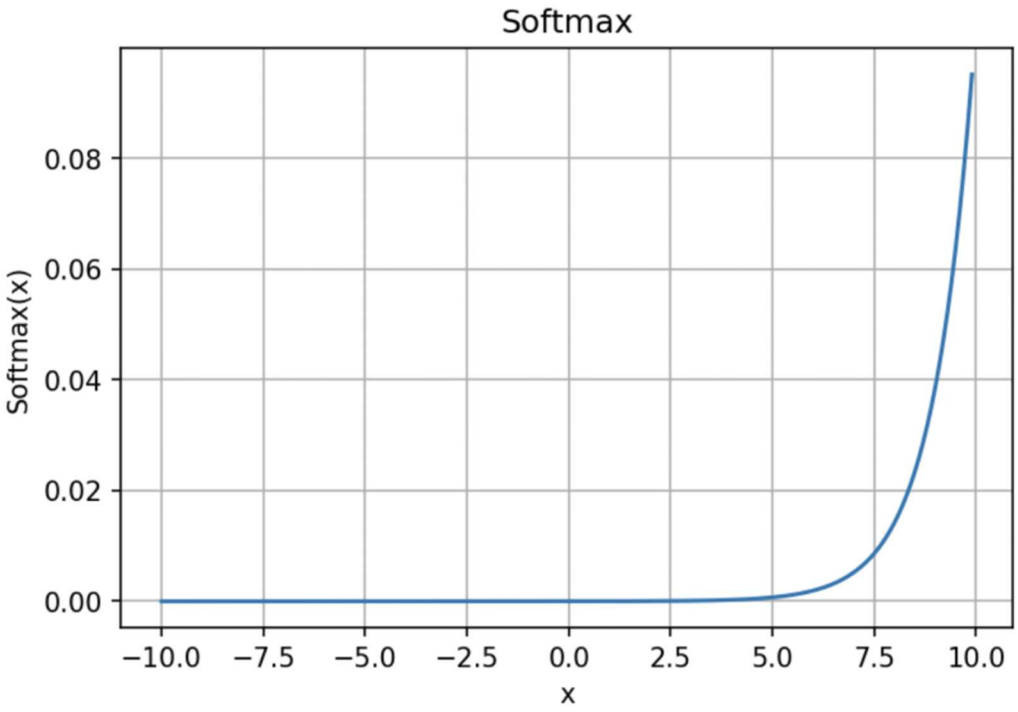
SOFTMAX

Softmax is an activation function that scales numbers/logits into probabilities. The output of a Softmax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes. The function is great for classification problems, especially if you're dealing with multi-class classification problems, as it will report back the "confidence score" for each class. Since we're dealing with probabilities here, the scores returned by the softmax function will add up to 1.

Many multi-layer neural networks end in a penultimate layer which outputs real-valued scores that are not conveniently scaled and which may be difficult to work with. Here the softmax is very useful because it converts the scores to a normalized probability distribution, which can be displayed to a user or used as input to other systems. For this reason, it is usual to append a softmax function as the final layer of the neural network.

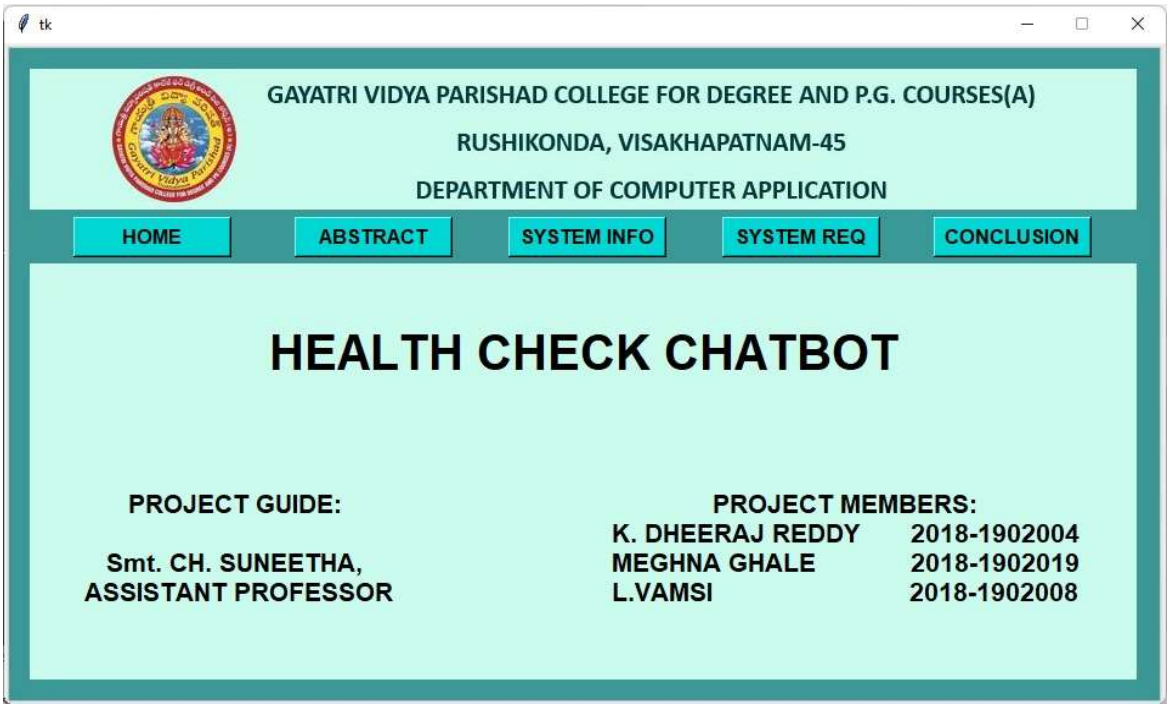
Mathematically,

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

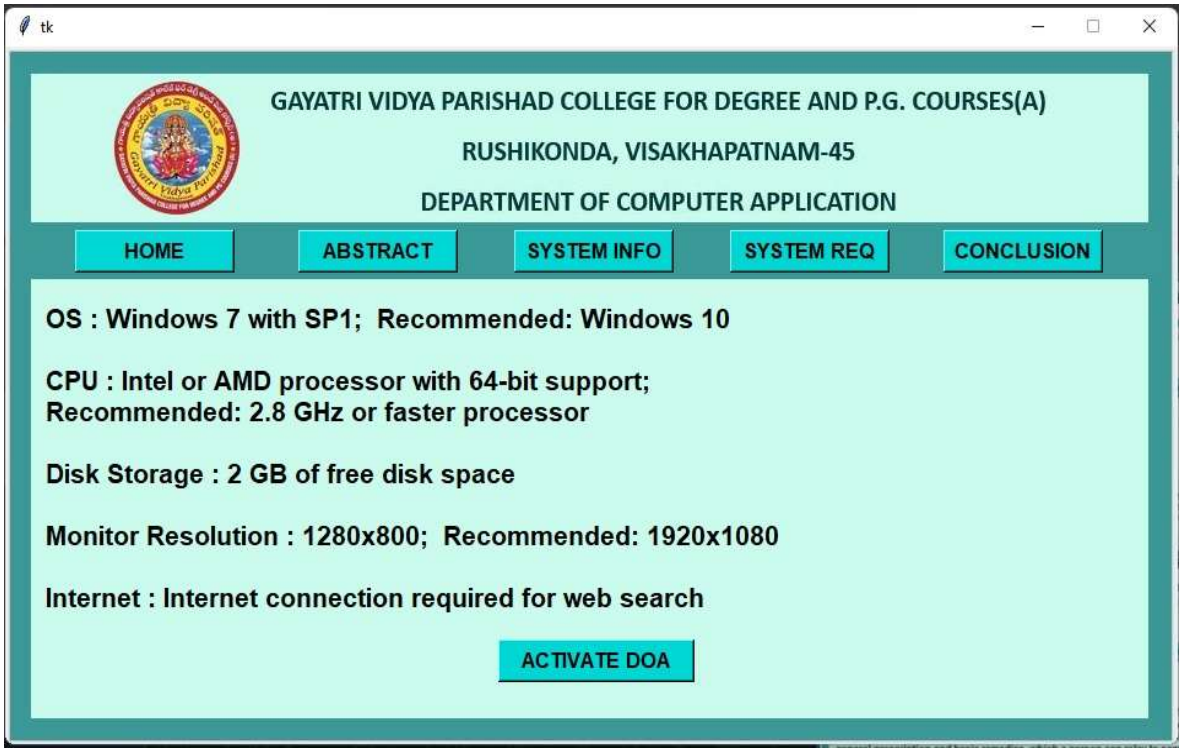


4.2 DESIGN EXAMPLE

MAIN PAGE DESIGN:



BOT ACTIVATION PAGE DESIGN:



USER QUERY: -

- **The chatbot is live**

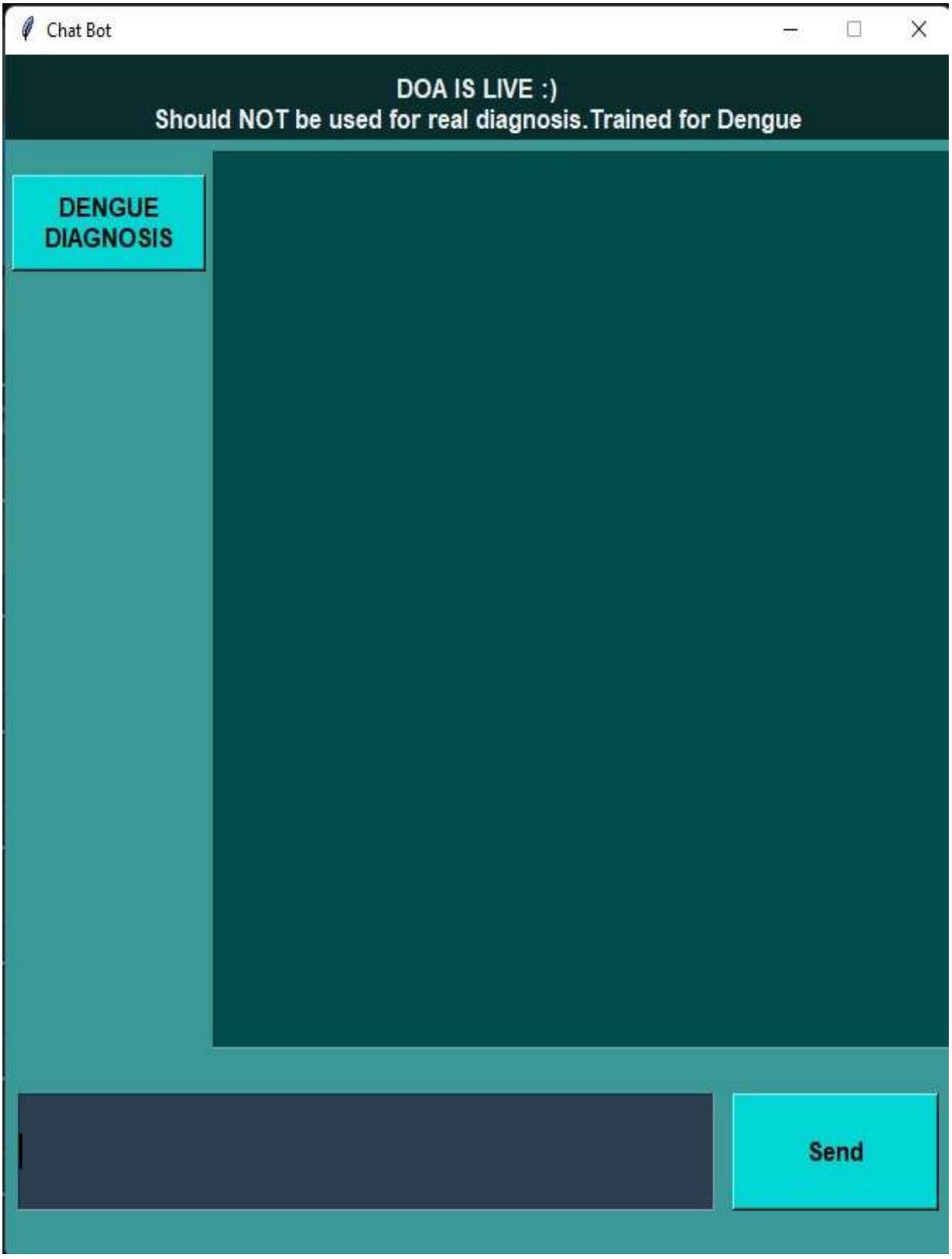


Fig 1: The Live chatbot

- **User enters query or starts the conversation**

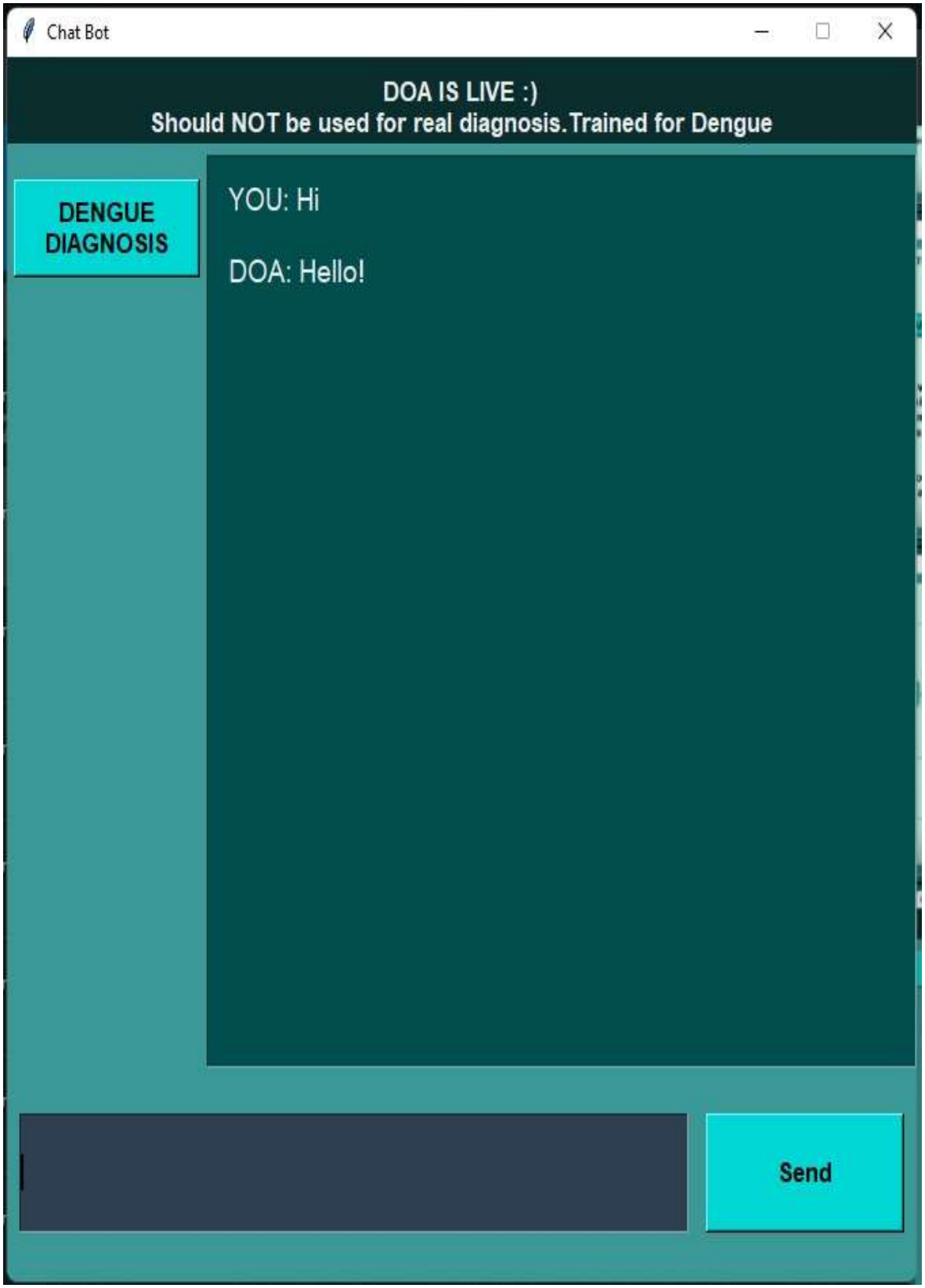


Fig 2: Conversation with chatbot

CODING

CODING

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost but the goal should be to reduce the cost of later phases. In other words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer.

5.1 CODING APPROACH

There are two major approaches for coding any software system. They are Top-Down approach and bottom-up approach.

Bottom-up Approach can best suit for developing the object-oriented systems. During system design phase of reduce the complexity. We decompose the system into an appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations.

Once objects have been modelled, they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other the Bottom-Up approach is more suitable for coding these objects. In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong.

5.2 INFORMATION HANDLING

Any software system requires some amount of information during its operation selection of appropriate data structures can help us to produce the code so that objects of the system can better operate with the available information decreased complexity.

In the current system we followed the coding rules for naming the variables and methods. As part of coding internal documentation is also provided that help the readers to better understand the code.

5.3 PROGRAMMING STYLE

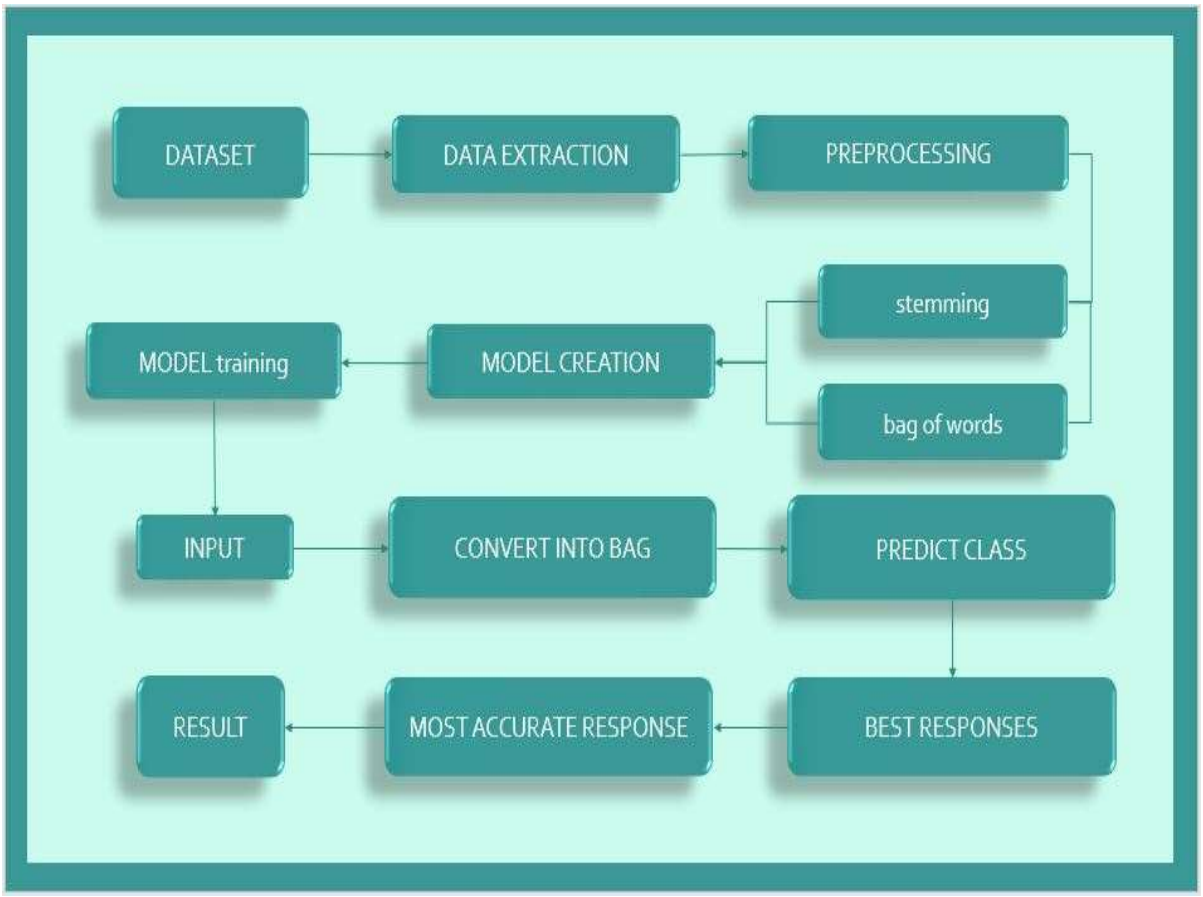
Programming style deals with act of rules that a programmer has to follow so that the characteristics of coding such as Traceability, Understandability, Modifiability, and Extensibility can be satisfied.

In the current system we followed the coding rules for naming the variables and methods. As part of coding internal documentation is also provided that help the readers to better understand the code.

5.4 VERIFICATION AND VALIDATION

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system Coding for the object has been thoroughly verified from different aspects regarding their design, in the way they are integrated and etc. The various techniques that have been followed for validation discussed in testing the current system.

5.5 FLOW OF THE CHATBOT:



5.6 SAMPLE SOURCE CODE

Model 1 coding

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy
import tflearn
import random
import json
from tkinter import *
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import webbrowser
from gui import *

#Loading JSON Data

with open("intents.json") as file:
    data = json.load(file)

    #Extracting Data

words = []
labels = []
docs_x = []
docs_y =[]
```

```
for intent in data["intents"]:
    for pattern in intent["patterns"]:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])

        if intent["tag"] not in labels:
            labels.append(intent["tag"])

    #Pre-Processing->

    #stemming
words = [stemmer.stem(w.lower())for w in words if w not in "?"]
words = sorted(list(set(words)))
labels = sorted(labels)

    #bag of words

training = []
output =[]

out_empty = [0 for _ in range(len(labels))]

for x, doc in enumerate(docs_x):
    bag = []

    wrds = [stemmer.stem(w) for w in doc]

    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)
```

```
    output_row = out_empty[:]
    output_row[labels.index(docs_y[x])] = 1

    training.append(bag)
    output.append(output_row)

training = numpy.array(training)
output = numpy.array(output)

    #model creation
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)

#training model

model.fit(training, output, n_epoch=500, batch_size=8, show_metric=True)

#Predictions(input, Convert a bag, prediction from the model, most probable class, Pick a
response the choosen class)
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]
    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1
```

```
    return numpy.array(bag)

bot_name = "DOA"

def get_response(msg):

    inp = msg

    results = model.predict([bag_of_words(inp, words)])[0]

    results_index = numpy.argmax(results)
    tag = labels[results_index]

    if results[results_index] > 0.7:
        for tg in data["intents"]:
            if tg['tag'] == tag:
                if 'google' in tag:
                        new=2
                        tabUrl="http://google.com/?#q="
                        webbrowser.open(tabUrl+tag,new=new)
                responses = tg['responses']

        return random.choice(responses)

    return "i didn't get that, try again"
```

Model 2 coding

```
import numpy
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
```

```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

from tkinter import *

BG_BLUE = "#0341fc"
BG_GRAY = "#3a9997"
BG_COLOR = "#004f4e"
TEXT_COLOR = "#EAECEE"
BT_COLOR = "#00d6d3"

#Loading CSV Data

Dataset=pd.read_csv("dengue.csv")

X = Dataset.values[:,:8]
Y = Dataset.values[:,-1]

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,random_state=2,test_size=0.25)

min_max_scaler = preprocessing.MinMaxScaler()
X_train=min_max_scaler.fit_transform(X_train)
X_test=min_max_scaler.transform(X_test)

#model creation --> model2

model2 = Sequential()

model2.add(Dense(32, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(32, activation='softmax'))
```

```
opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)
#can use mse
model2.compile(loss='sparse_categorical_crossentropy',
          optimizer=opt,
          metrics=['accuracy'])

#training model2

model2.fit(X_train, Y_train, epochs=150, validation_data=(X_test,Y_test))
root=Tk()
root.title("DENGUE")
root.resizable(width=False, height=False)
root.configure(bg=BG_GRAY)



frame=Frame(root)
frame.configure(bg=BG_GRAY)

arr=[]

def destr():
   output.configure(state='normal')
   output.delete(1.0,END)

def but():
   a=F.get()
   if a=="YES" or a=="yes":
      arr.append(1)
   else:
      arr.append(0)
   F.delete(0,END)

   a=H.get()
   if a=="YES" or a=="yes":
```

```
    arr.append(1)
else:
    arr.append(0)
H.delete(0,END)

a=B.get()
if a=="YES" or a=="yes":
    arr.append(1)
else:
    arr.append(0)
B.delete(0,END)

a=J.get()
if a=="YES" or a=="yes":
    arr.append(1)
else:
    arr.append(0)
J.delete(0,END)

a=T.get()
if a=="YES" or a=="yes":
    arr.append(1)
else:
    arr.append(0)
T.delete(0,END)

a=A.get()
if a=="YES" or a=="yes":
    arr.append(1)
else:
    arr.append(0)
A.delete(0,END)

a=AP.get()
```

```
    if a=="YES" or a=="yes":
        arr.append(1)
    else:
        arr.append(0)
    AP.delete(0,END)

    a=V.get()
    if a=="YES" or a=="yes":
        arr.append(1)
    else:
        arr.append(0)
    V.delete(0,END)

    results2 = model2.predict(numpy.array([arr], dtype=numpy.float32))[0]
    print(results2)
    results2_index = numpy.argmax(results2)

    output.configure(state='normal')
    output.insert(END, "\t\tMedical Prescription")
    output.configure(state='disabled')

    if results2[results2_index] > 0.75:
        output.configure(state='normal')
        output.insert(END, "\n\n\nYou have dengue symptoms\n\nPlease consult a infectious
diseases specialist ASAP\n\nFirst Aid Medication:\n\nPyrapem syrup -> 15ml\n=>after meal
3 times a day\nCalpal ->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter dinner
1\nCaripill ->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter dinner 1\nInjection ->
Montaz")
        output.configure(state='disabled')

        del arr[:]

    else:
        output.configure(state='normal')
```

```
        output.insert(END, "\n\n\nYou don't have dengue")
        output.configure(state='disabled')

        if arr[0] == 1:
            output.configure(state='normal')
            output.insert(END, "\n\nmedicine for fever:\nCalphal ->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter Dinner 1\nDolo->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter dinner 1")
            output.configure(state='disabled')

        if arr[1] == 1:
            output.configure(state='normal')
            output.insert(END, "\n\nmedicine for headache:\nCalphal ->2 Tablets(2 Days course)\n=>After Lunch 1\nNeproshin D->2 Tablets(2 Days course)\n=>After Lunch 1")
            output.configure(state='disabled')

        if arr[2] == 1:
            output.configure(state='normal')
            output.insert(END, "\n\nmedicine for body pains:\nFastac plus ->4 Tablets(2 Days course)\n=>After Lunch 1\nAfter Dinner 1\nZerodol P ->2 Tablets(1 Day course)\n=>After Lunch 1\nAfter Dinner 1\nInjection -> Zonac")
            output.configure(state='disabled')

        if arr[3] == 1:
            output.configure(state='normal')
            output.insert(END, "\n\nmedicine for joint pains: \nAcelo Plus ->4 Tablets(2 Days course)\n=>After Breakfast 1\nAfter Dinner 1\nAcelodol MR ->4 Tablets(2 Days course)\n=>After Breakfast 1\nAfter Dinner 1")
            output.configure(state='disabled')

        if arr[4] == 1:
            output.configure(state='normal')
```

```
        output.insert(END, "\n\nmedicine for taste loss: \nAmitriptyline ->4 Tablets(2 Days course)\n=>After Breakfast 1\nAfter Dinner 1\nMetronidazole ->4 Tablets(2 Days course)\n=>After Breakfast 1\nAfter Dinner 1")
        output.configure(state='disabled')

    if arr[5] == 1:
        output.configure(state='normal')
        output.insert(END, "\n\nmedicine and dite for  appetite loss:")
        output.configure(state='disabled')

    if arr[6] == 1:
        output.configure(state='normal')
        output.insert(END, "\n\nmedicine for abdominal pain:\nCylophom ->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter dinner 1\nBigspas ->4 Tablets(2 Days course)\n=>After breakfast 1\nAfter dinner 1\nAnaspas ->Afternoon 1 Tablet")
        output.configure(state='disabled')

    if arr[7] == 1:
        output.configure(state='normal')
        output.insert(END, "\n\nmedicine for vomiting:\nVomikind ->2 Tablets(1 Day course)\n=>Morning 1\nAfter Dinner 1\nPerinorm ->2 Tablets(1 Day course)\n=>Morning 1\nAfter dinner 1\nInjection -> Vomikind")
        output.configure(state='disabled')

    output.configure(state='normal')
    output.insert(END, "\n\n\nIf your symptoms are getting worse, consult a infectious\ndiseases specialist \n")
    output.configure(state='disabled')

  del arr[:]

HEAD=Label(frame,
     text="ENTER YES IF YOU ARE HAVE THE SYMPTOM \nAND\n ENTER NO IF YOU ARE NOT HAVE THE SYMPTOM",
```

```
        font=("calibri", 18,'bold'),
        bg=BG_GRAY)
HEAD.grid(row=0, column=0,columnspan=2,padx=30, pady=10)

OUT=Label(frame,
        text="DIAGNOSIS RESULT",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY)
OUT.grid(row=0, column=2,padx=50, pady=10)
output=Text(frame,
       width=50,
       height=28,
       fg="#ffffff",
       bg=BG_COLOR,
       font=("calibiri", 10), state='disabled')
output.grid(row=1,column=2,rowspan=10,padx=15)

FEVER=Label(frame,
        text="Are you having fever ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,
        justify="left")
FEVER.grid(row=2, column=0,padx=10)

F=Entry(frame,
       width=20,
       font=("calibiri", 14,'bold'),
       fg="#022423",
       bg="#0bb3af")
F.grid(row=2, column=1, pady=10,padx=50)

HEADACHE=Label(frame,
        text="Are you having headache ",
        font=("calibri", 18,'bold'),
```

```
        bg=BG_GRAY,
        justify="left")
HEADACHE.grid(row=3, column=0,padx=10)

H=Entry(frame, width=20, font=("calibiri", 14,'bold'),fg="#022423",bg="#0bb3af")
H.grid(row=3, column=1, pady=10,padx=50)

body_pains=Label(frame,
        text="Are you having body pains ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,
        justify="left")
body_pains.grid(row=4, column=0,padx=10)

B=Entry(frame,
       width=20,
       font=("calibiri", 14,'bold'),
       fg="#022423",
       bg="#0bb3af')
B.grid(row=4, column=1, pady=10,padx=50)

joint_pains=Label(frame,
        text="Are you having joint pains ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,
        justify="left")
joint_pains.grid(row=5, column=0,padx=10)

J=Entry(frame,
       width=20,
       font=("calibiri", 14,'bold'),
       fg="#022423",
       bg="#0bb3af')
J.grid(row=5, column=1, pady=10,padx=50)
```

```
taste=Label(frame,
        text="Are you not able to taste ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,
        justify="left")
taste.grid(row=6, column=0,padx=10)

T=Entry(frame,
       width=20,
       font=("calibiri", 14,'bold'),
       fg="#022423",
       bg="#0bb3af")
T.grid(row=6, column=1, pady=10,padx=50)

appetite_loss=Label(frame,
        text="Appetite loss ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,  justify="left")
appetite_loss.grid(row=7, column=0,padx=10)

A=Entry(frame,
       width=20,
       font=("calibiri", 14,'bold'),
       fg="#022423",
       bg="#0bb3af")
A.grid(row=7, column=1, pady=10,padx=50)

abdominal_pain=Label(frame,
        text="Are you having Abdominal pain ",
        font=("calibri", 18,'bold'),
        bg=BG_GRAY,
        justify="left")
abdominal_pain.grid(row=8, column=0,padx=10)
```

```
AP=Entry(frame,
        width=20,
        font=("calibiri", 14,'bold'),
        fg="#022423",
        bg="#0bb3af")
AP.grid(row=8, column=1, pady=10,padx=50)
vomiting=Label(frame,
         text="Are you having nausea OR vomiting ",
         font=("calibri", 18,'bold'),
         bg=BG_GRAY,
         justify="left")
vomiting.grid(row=9, column=0,padx=10)

V=Entry(frame,
        width=20,
        font=("calibiri", 14,'bold'),
        fg="#022423",
        bg="#0bb3af")
V.grid(row=9, column=1, pady=10,padx=50)

FBUTTON=Button(frame,
      text="START DIAGNOSIS",
      width=15,
      font=("lora", 14,'bold'),
      bg=BT_COLOR,
      command=but)
FBUTTON.grid(row=11, column=0,columnspan=2,pady=40)

dest_button=Button(frame,
      text="CLEAR RESULT",
      width=13,
      font=("lora", 14,'bold'),
      bg=BT_COLOR,
      command=destr)
```

```
dest_button.grid(row=11, column=2,columnspan=2,pady=40)

frame.pack()
root.mainloop()
```

Chatbot GUI coding

```
from tkinter import *
from main import *
import os

BG_BLUE = "#0341fc"
BG_GRAY = "#3a9997"
BG_COLOR = "#004f4e"
TEXT_COLOR = "#EAECEE"
BT_COLOR = "#00d6d3"

FONT = "Helvetica 14"
FONT_BOLD = "Helvetica 13 bold"
def chat2():
    os.system('python model2.py')

class ChatApplication:

    def __init__(self):
        self.window = Tk()
        self._setup_main_window()

    def run(self):
        self.window.mainloop()
```

```
    def _setup_main_window(self):
        self.window.title("Chat Bot")
        self.window.resizable(width=False, height=False)
        self.window.configure(width=670, height=750, bg=BG_COLOR)

        # head label
        head_label = Label(self.window, bg="#092e2d", fg=TEXT_COLOR,
                           text="DOA IS LIVE :)\nShould NOT be used for real diagnosis.Trained for
Dengue", font=FONT_BOLD, pady=10)
        head_label.place(relwidth=1)

        # tiny divider
        line = Label(self.window, width=450, bg=BG_GRAY)
        line.place(relwidth=1, rely=0.07, relheight=0.012)

        # text widget
        self.text_widget = Text(self.window, width=20, height=2, bg=BG_COLOR,
fg=TEXT_COLOR,
                                font=FONT, padx=15, pady=15)
        self.text_widget.place(relheight=0.745, relwidth=0.780,relx=0.220, rely=0.08)
        self.text_widget.configure(cursor="arrow", state=DISABLED)

        #selection table
        r_label = Label(self.window, bg=BG_GRAY, height=40, width=50)
        r_label.place(relwidth=0.220, rely=0.08)

        #dengue button

        dengue_button = Button(r_label, text="DENGUE\nDIAGNOSIS", font=FONT_BOLD,
width=20, bg=BT_COLOR, command=chat2)
        dengue_button.place(relx=0.02, rely=0.022, relheight=0.10, relwidth=0.960)

        # bottom label
        bottom_label = Label(self.window, bg=BG_GRAY, height=80)
```

```
        bottom_label.place(relwidth=1, rely=0.825)

        # message entry box
        self.msg_entry = Entry(bottom_label, bg="#2C3E50", fg=TEXT_COLOR, font=FONT)
        self.msg_entry.place(relwidth=0.74, relheight=0.06, rely=0.022, relx=0.011)
        self.msg_entry.focus()
        self.msg_entry.bind("<Return>", self._on_enter_pressed)

        # send button
        send_button = Button(bottom_label, text="Send", font=FONT_BOLD, width=20,
bg=BT_COLOR, command=lambda: self._on_enter_pressed(None))
        send_button.place(relx=0.77, rely=0.022, relheight=0.06, relwidth=0.22)
    def _on_enter_pressed(self, event):
        msg = self.msg_entry.get()
        self._insert_message(msg, "YOU")
    def _insert_message(self, msg, sender):
        if not msg:
            return
        self.msg_entry.delete(0, END)
        msg1 = f"{sender}: {msg}\n\n"
        self.text_widget.configure(state=NORMAL)
        self.text_widget.insert(END, msg1)
        self.text_widget.configure(state=DISABLED)
        msg2 = f"{bot_name}: {get_response(msg)}\n\n"

        self.text_widget.configure(state=NORMAL)
        self.text_widget.insert(END, msg2)
        self.text_widget.configure(state=DISABLED)
self.text_widget.see(END)

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

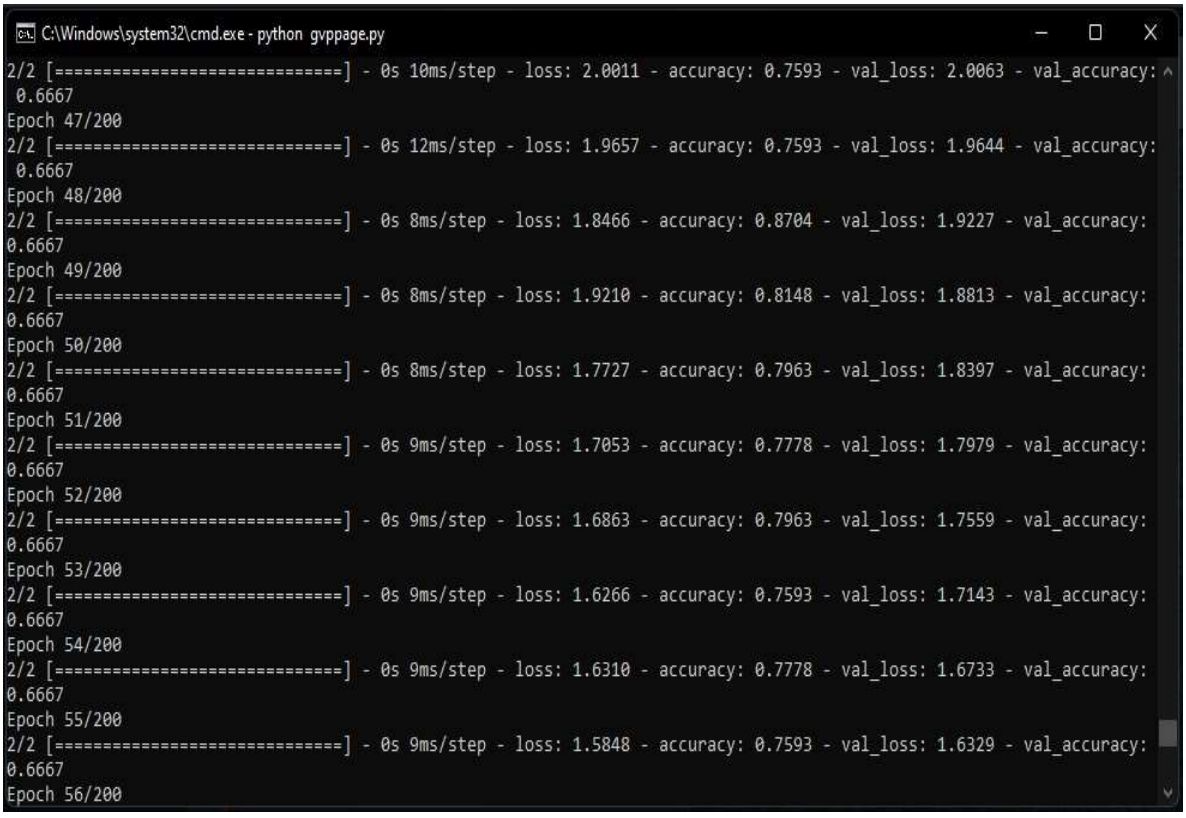
TRAINING

TRAINING

6.1 TRAINING MODEL 1

```
        #model creation
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)

#training model

model.fit(training, output, n_epoch=500, batch_size=8, show_metric=True)
```

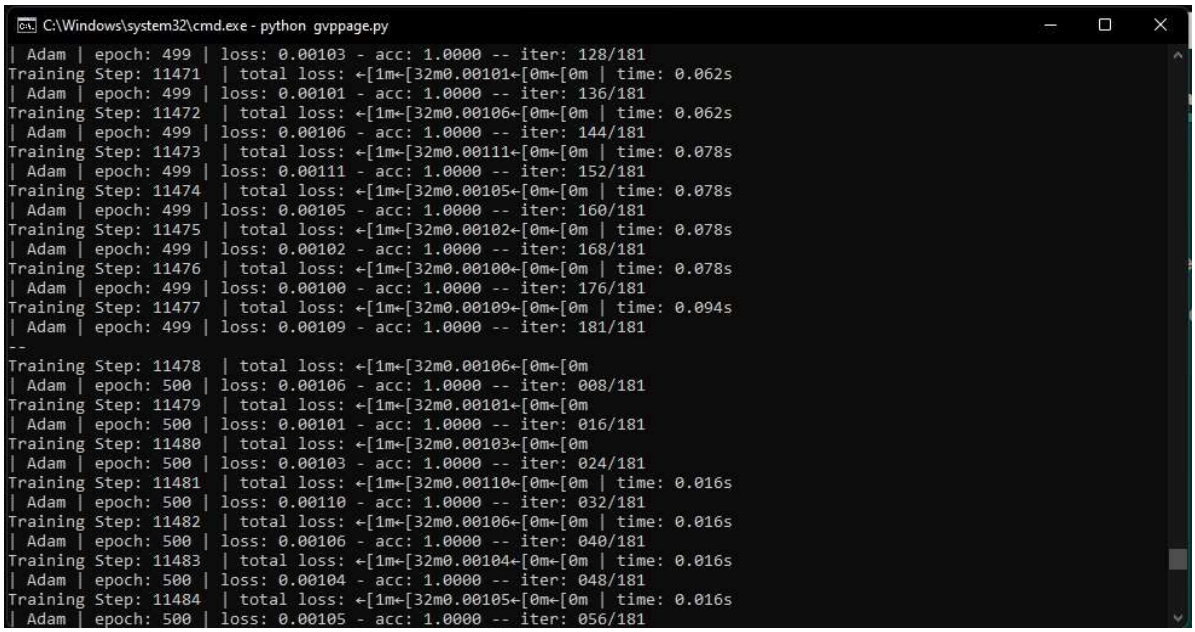
6.2 TRAINING CHATBOT



6.3 TRAINING MODEL 2

```
#model creation --> model2

model2 = Sequential()

model2.add(Dense(32, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(32, activation='softmax'))

opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)
#can use mse
model2.compile(loss='sparse_categorical_crossentropy',
        optimizer=opt,
        metrics=['accuracy'])

#training model2

model2.fit(X_train, Y_train, epochs=150, validation_data=(X_test,Y_test))
```

6.4 TRAINING DENGUE DIAGNOSIS



TESTING

TESTING

Testing is the process of finding differences between the expected behaviour specified by system models and the observed behaviour of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the code so the application should be thoroughly tested and validated.

Testing is the process of finding differences between the expected behaviour specified by system models and the observed behaviour of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the code so the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and a subset of integrated subsystems. Functional testing finds differences between the use case model and the system.

Finally, performance testing, finds differences between non-functional requirements and actual system performance. From modelling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

Different levels of testing are as follows:

Unit Testing:

In this test each component is tested independent of the other thus allowing parallelism in testing activity.

Integration Testing:

Many tested modules are combined into subsystem. The emphasis is on testing interface between modules. This activity can be considered to test the design.

System Testing:

This is a series of testing whose purpose is to fully exercise the entire software system. System testing verifies that all elements mesh properly and the overall system function/performance is achieved. System testing is done using real data. The system performance was found to be working to be as per requirements specified.

System Testing:

- System testing ensures that the complete system compiles with the functional requirements and non-functional requirements of the system, the following are some system testing activities.
- Functional testing finds differences between the functional between the functional requirements and the system. Test cases are divided from the use case model.
- Performance testing finds differences between the design and the system the design goals are derived from the functional requirements.
- Pilot testing the system is installed and used by a selected set of users – users exercise the system as if it had been permanently installed.
- Acceptance testing, I have followed benchmarks testing in a benchmarks testing the client prepares a set of test cases represent typical conditions under which the system operates. In our project, there are no existing benchmarks.

7.1 TESTING ACTIVITIES

Testing a large system is a complex activity and like any complex activity. It has to be broke into smaller activities. Thus, incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to form the subsystem for system testing.

7.2 TESTING PLAN

Testing accounts for 45 - 75% of the typical project effort. It is also one of the most commonly underestimated activities on a project. A test plan is a document that answers the basic questions about your testing effort. It needs to be initiated during the requirements gathering phase of the project and should evolve into a roadmap for the testing phase.

- Test Planning enables a more reliable estimate of the testing effort up front.
- It allows the project team time to consider ways to reduce the testing effort without being under time pressure.
- Test Plan helps to identify problem areas and focuses the testing team's attention on the critical paths.
- Test plan reduces the probability of implementing non-tested components.

7.3. TESTING MODEL 1

- **The chatbot is live**

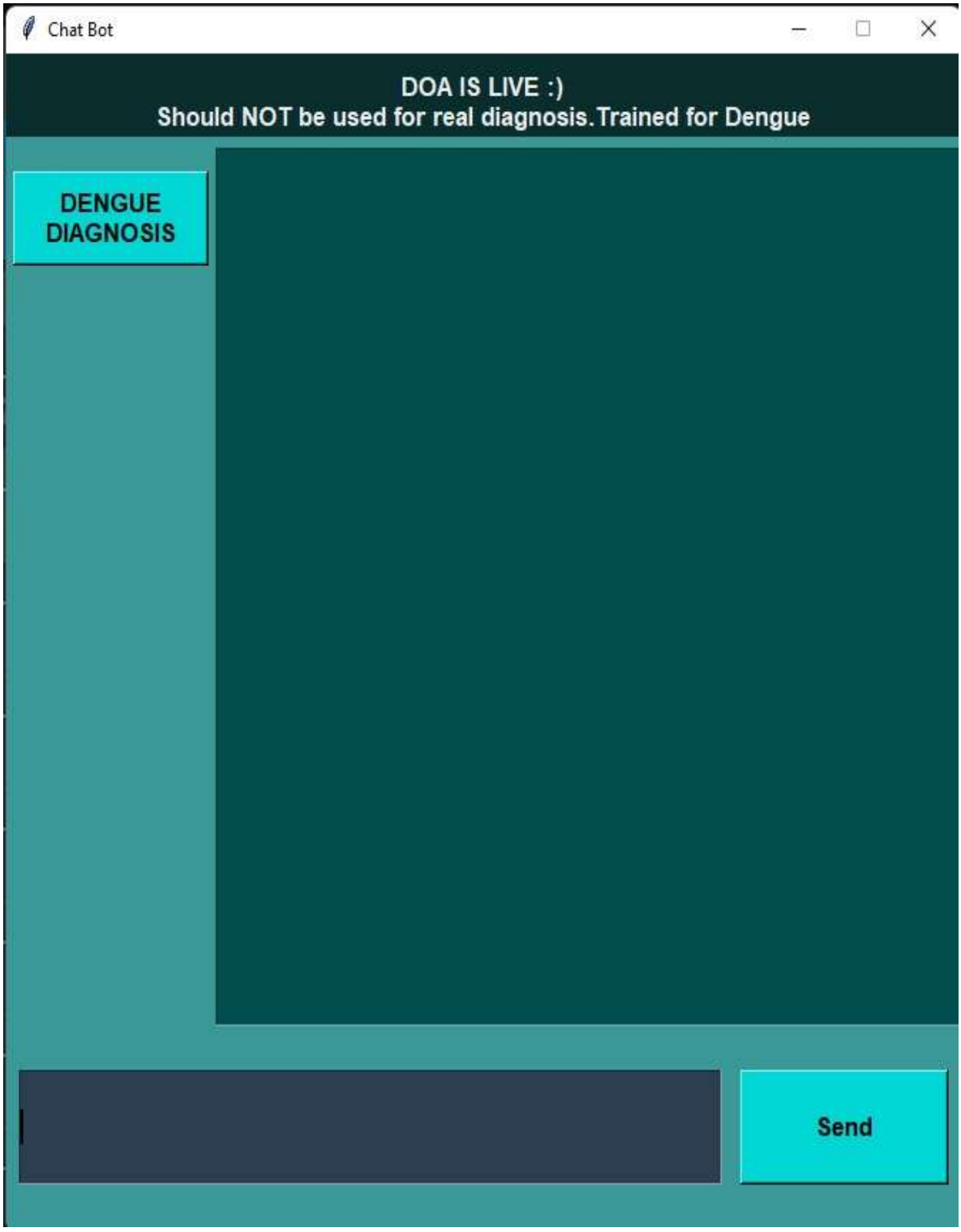


Fig 1: The Live chatbot

7.4. TESTING CHATBOT

- **User enters query or starts the conversation**

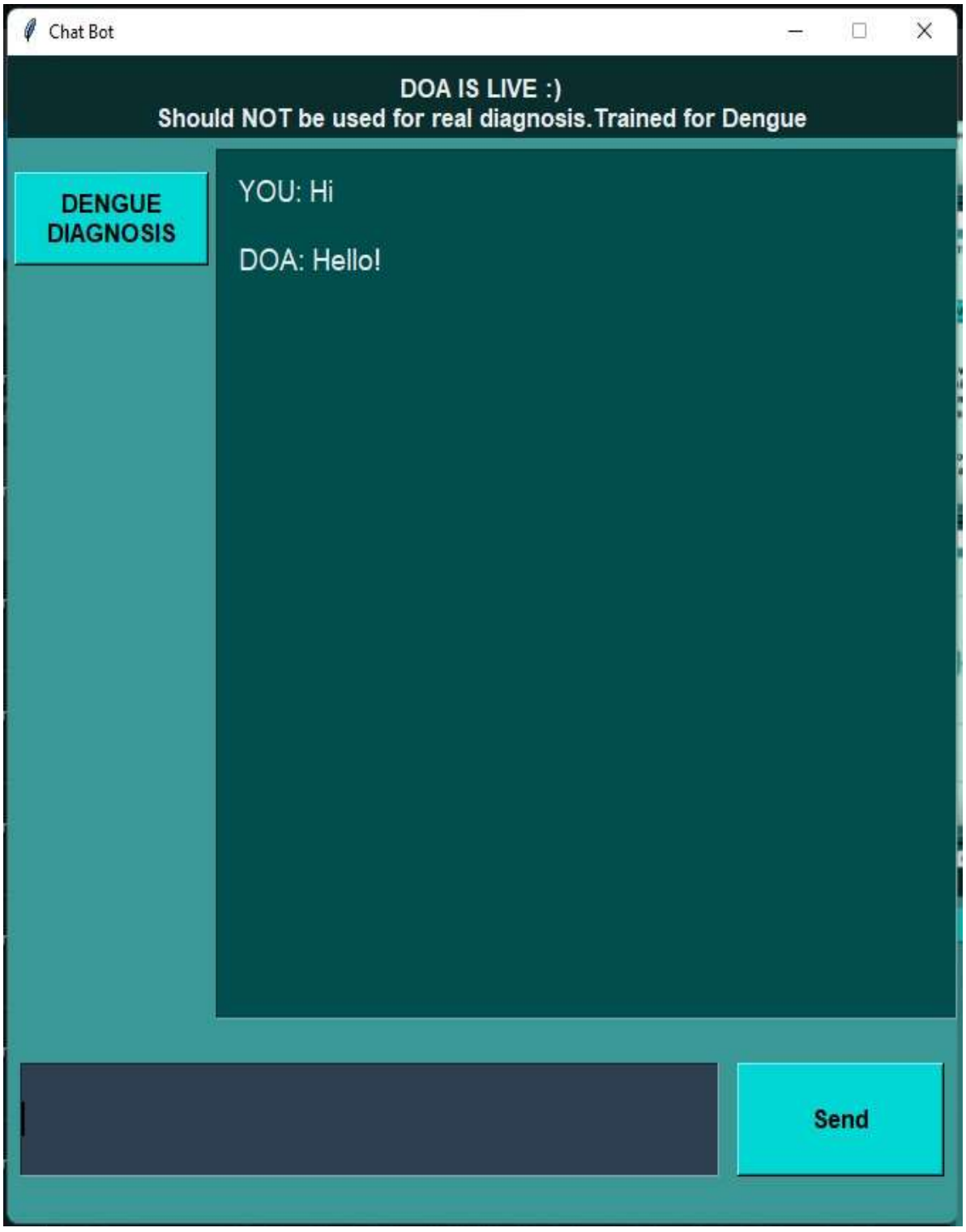


Fig 2: Conversation starts

- **Dengue Diagnosis Button**

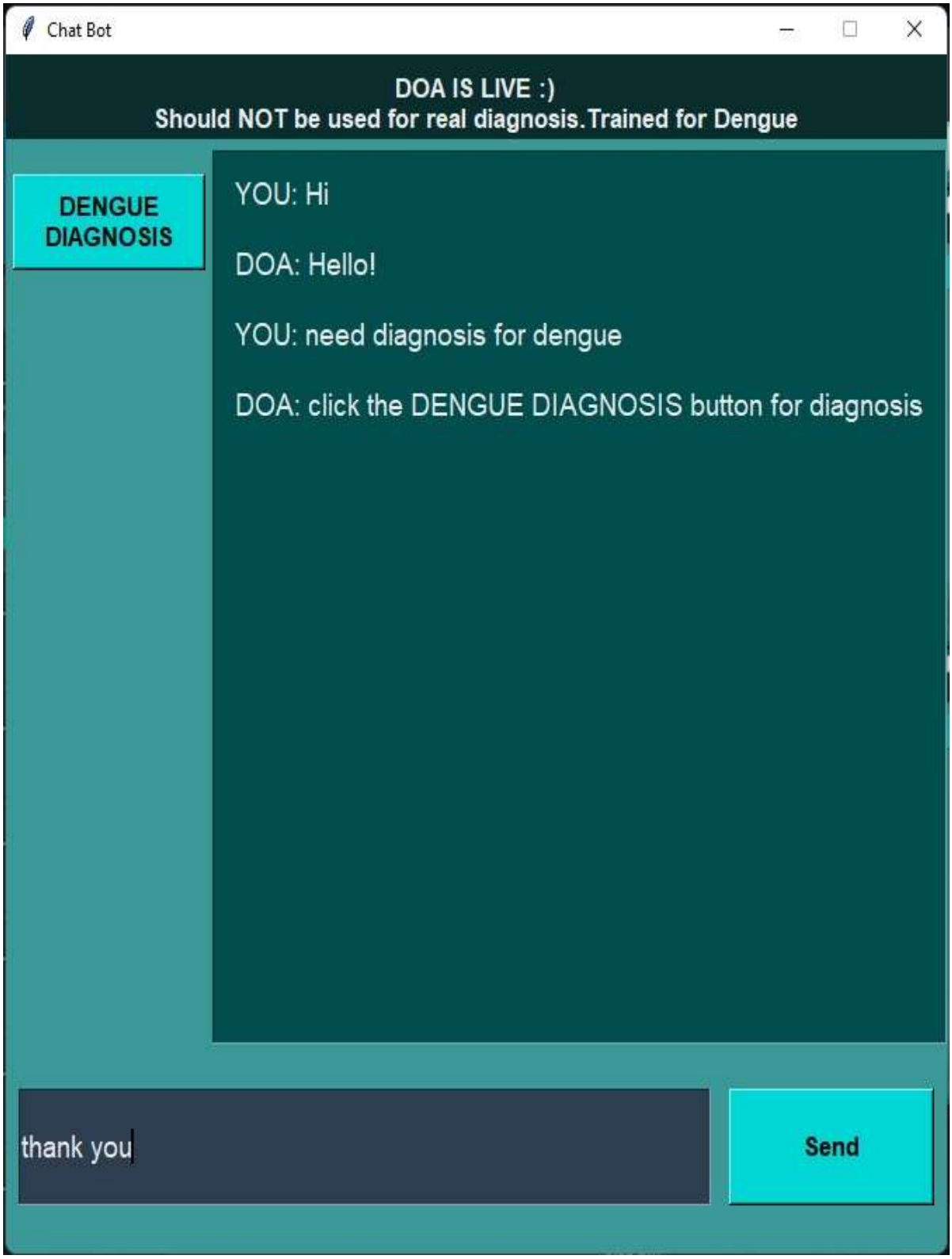


Fig 3: dengue diagnosis query

IF does not match the query:

- **It goes back to the start of the conversation**

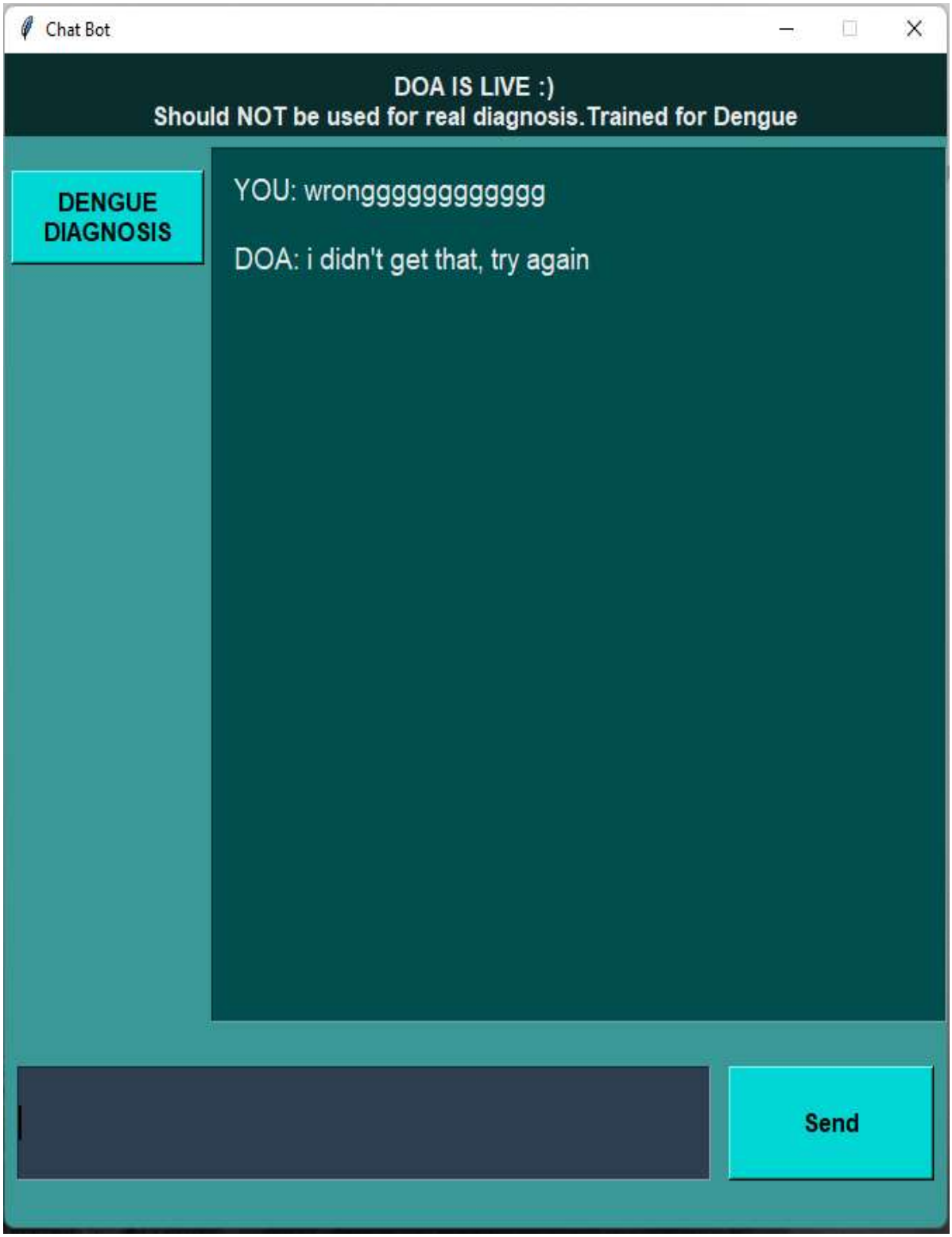


Fig 3: wrong query

7.5. TESTING MODEL 2

- **The dengue diagnosis button is clicked and opened**

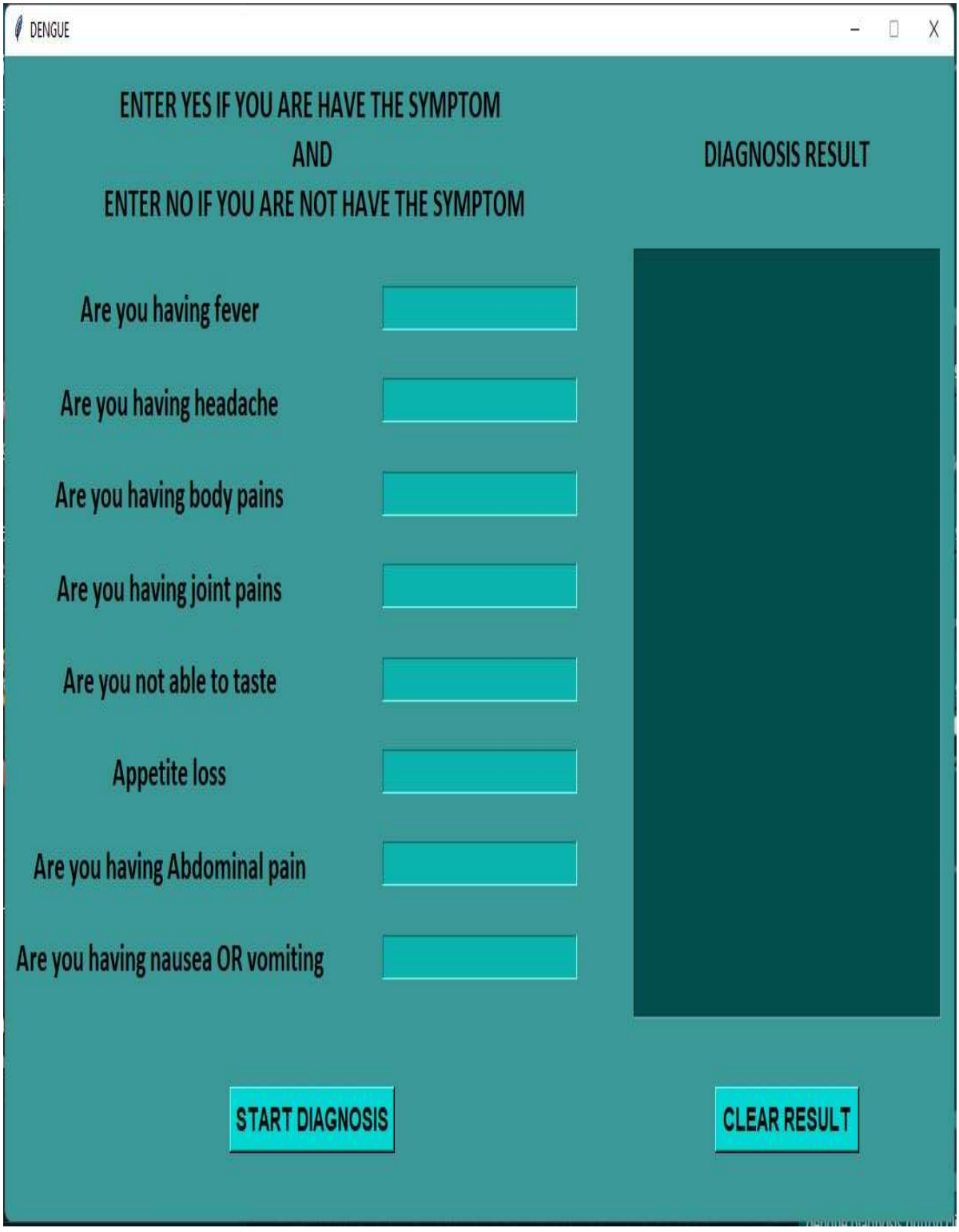


Fig. 3: start diagnosis

7.6 TESTING DENGUE DIAGNOSIS

- **User enters the diagnosis query**

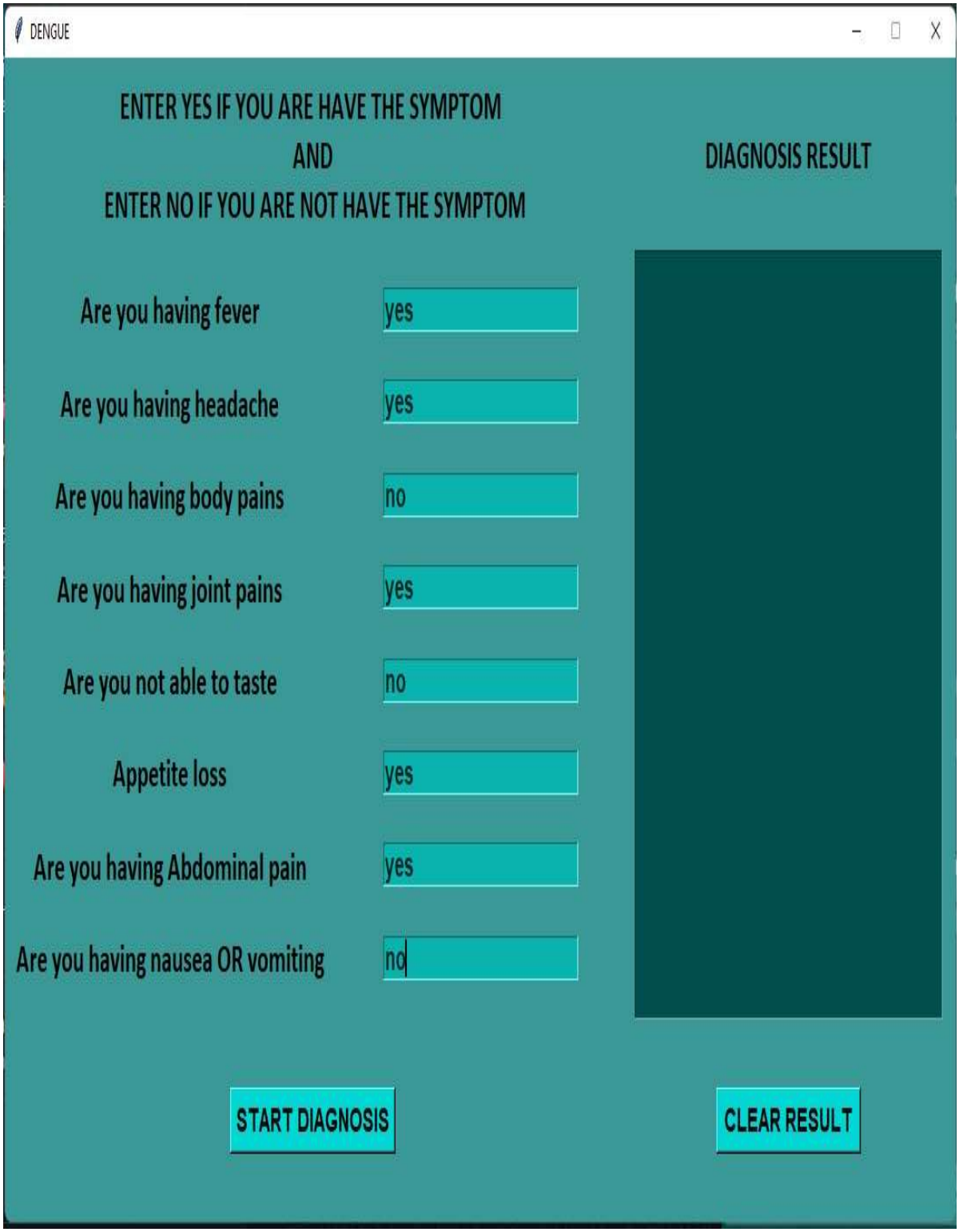


Fig 4: User's entry

- **The result of the user entered query**

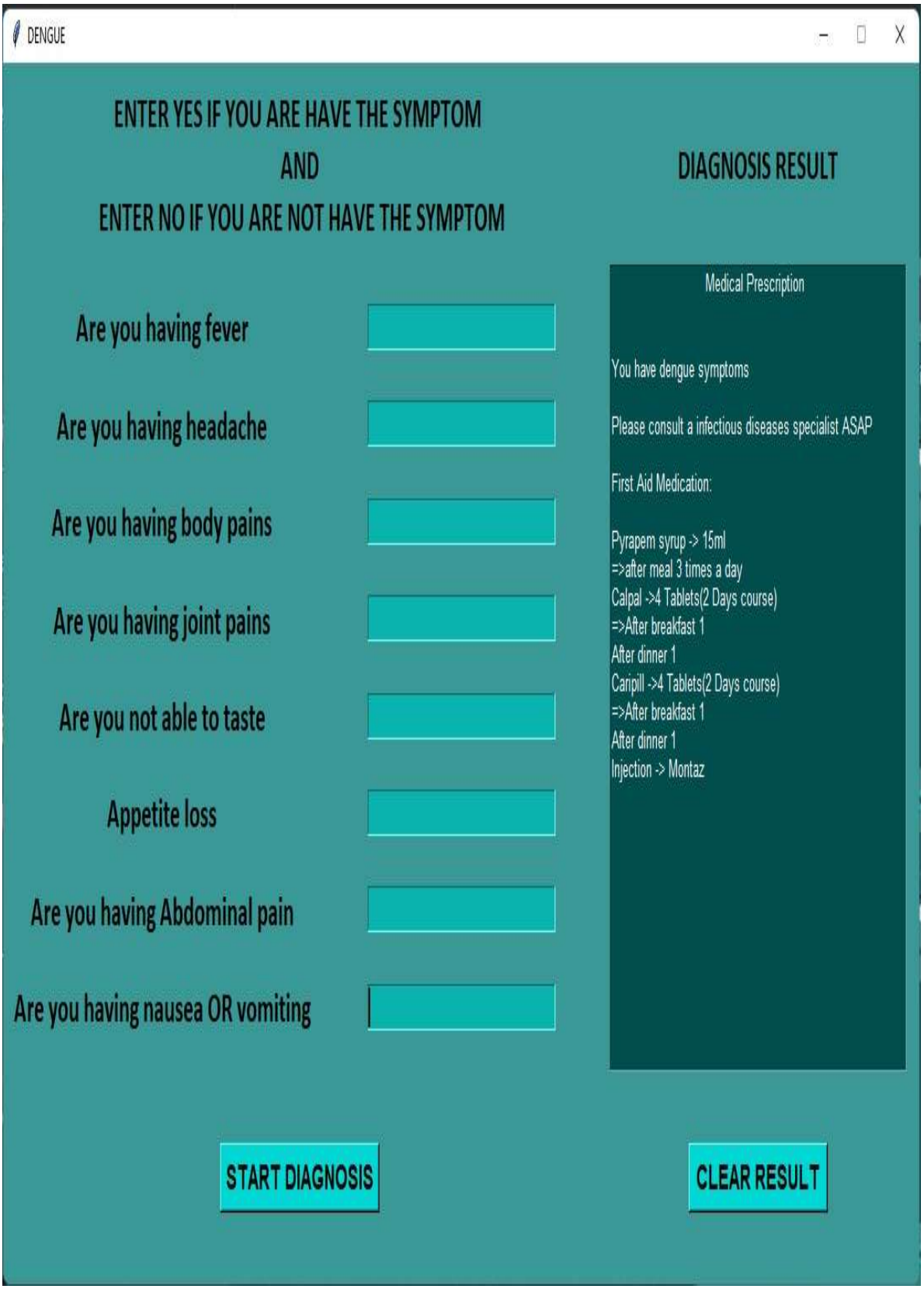


Fig 5: Diagnosis result (medical prescription)

7.7 TEST CASE REPORT

Report 1:

FEVER	= NO
HEADACHE	= YES
BODY PAINS	= NO
JOINT PAINS	= YES
TASTE LOSS	= NO
APPETITE LOSS	= NO
ABDOMINAL PAIN	= NO
NAUSEA	= YES

ANN RESULT

DIAGNOSIS RESULT IS : NEGATIVE

Report 2:

FEVER	= YES
HEADACHE	= YES
BODY PAINS	= YES
JOINT PAINS	= YES
TASTE LOSS	= NO
APPETITE LOSS	= YES
ABDOMINAL PAIN	= YES
NAUSEA	= YES

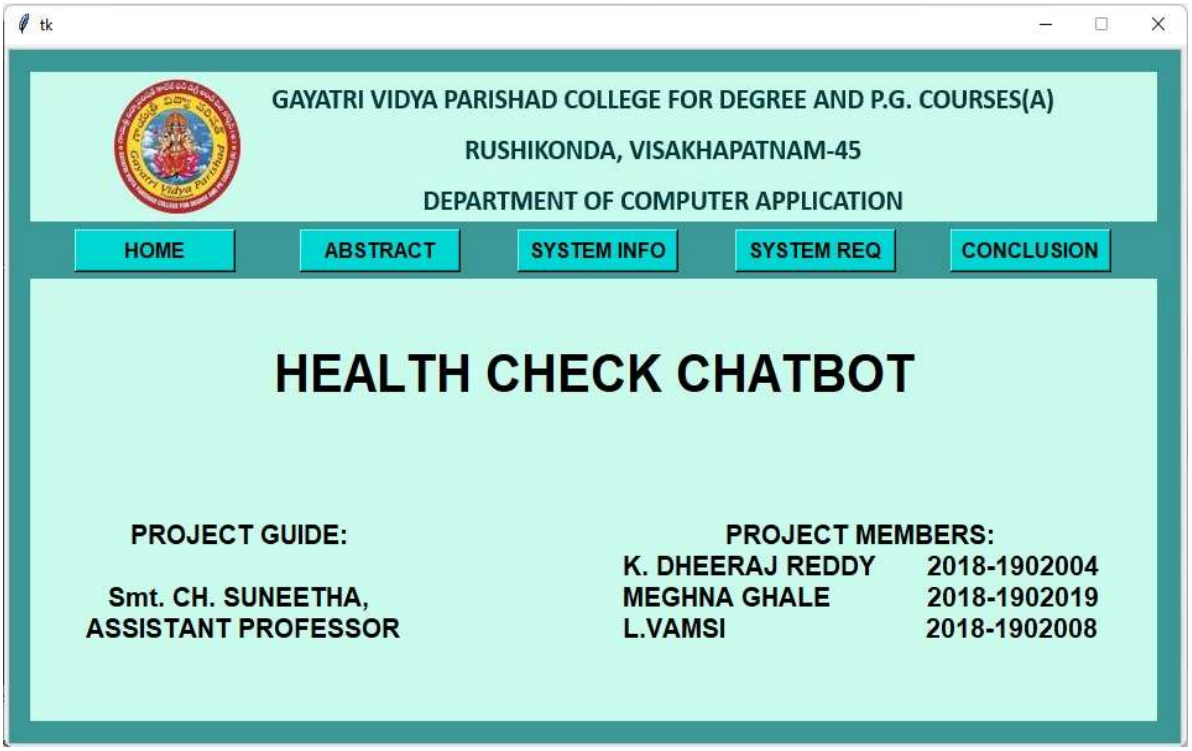
ANN RESULT

DIAGNOSIS RESULT IS : POSITIVE

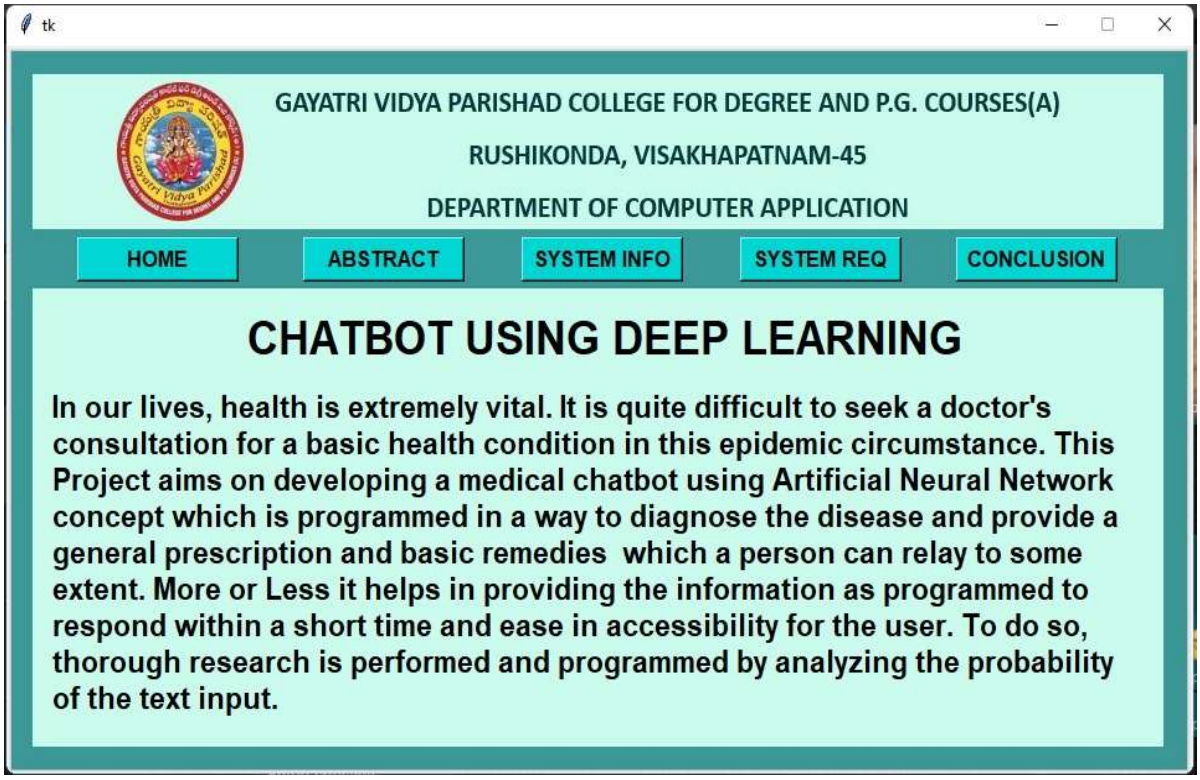
RESULT

RESULT

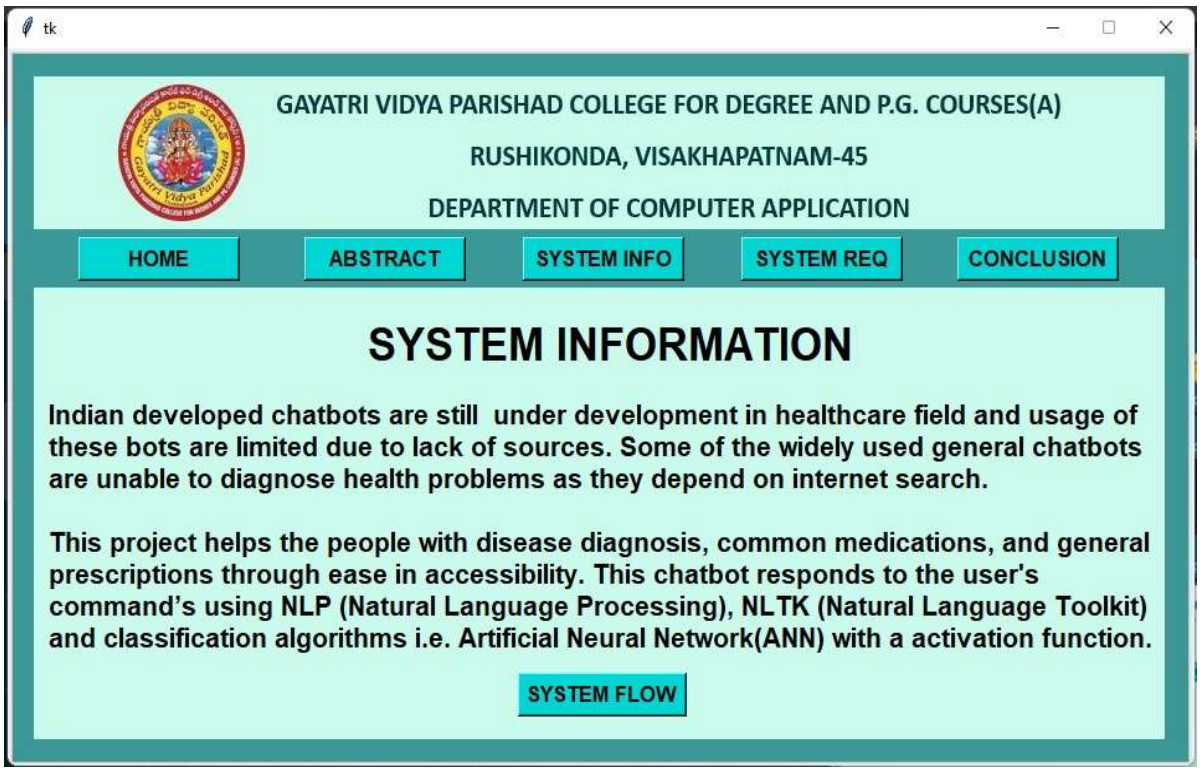
HOME



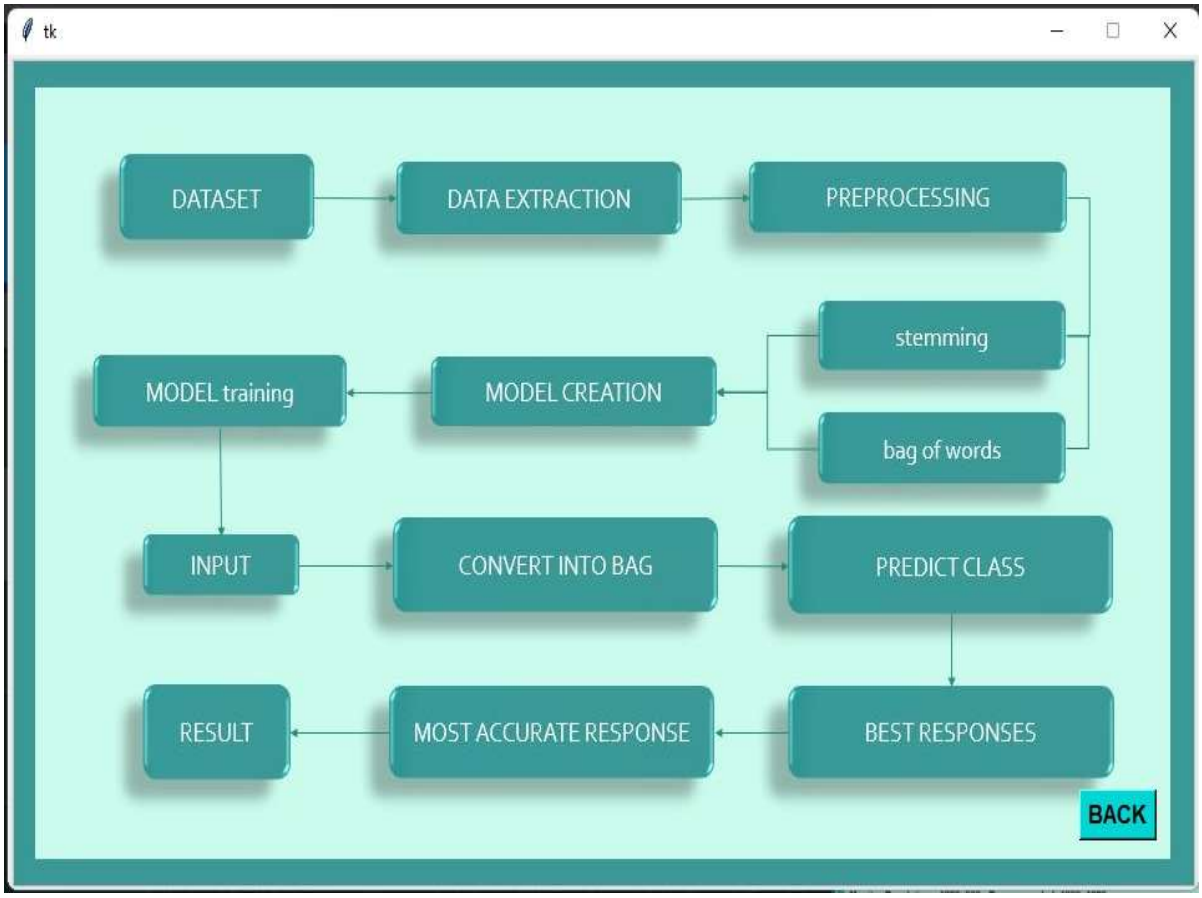
ABSTRACT



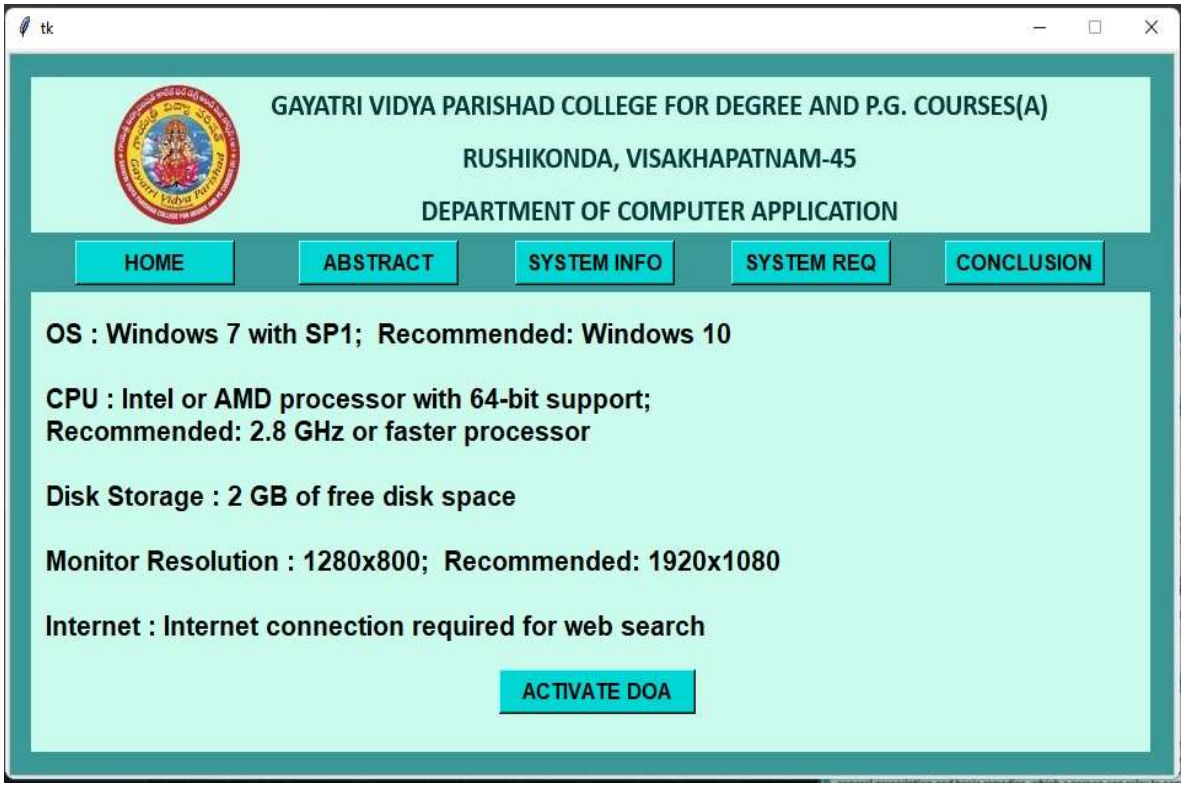
SYSTEM INFORMATION



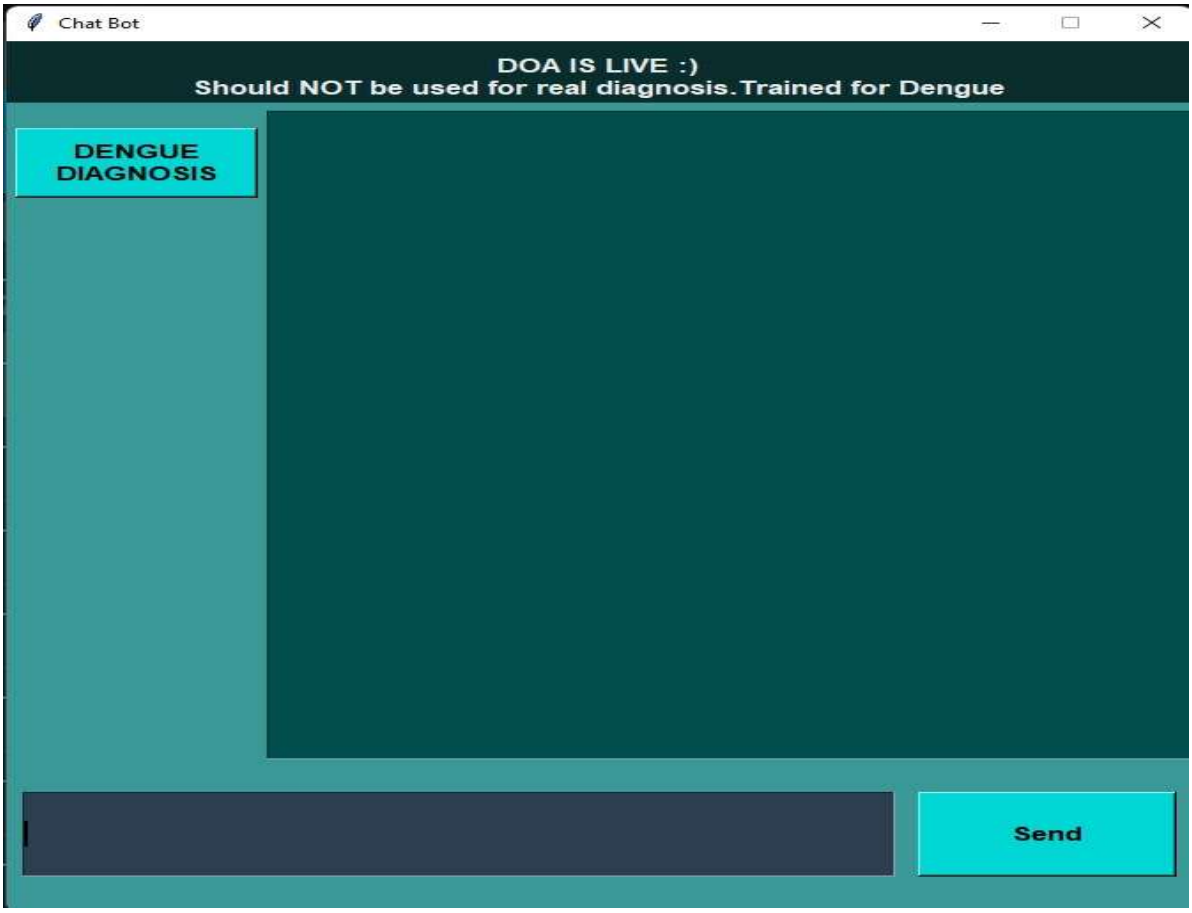
SYSTEM FLOW



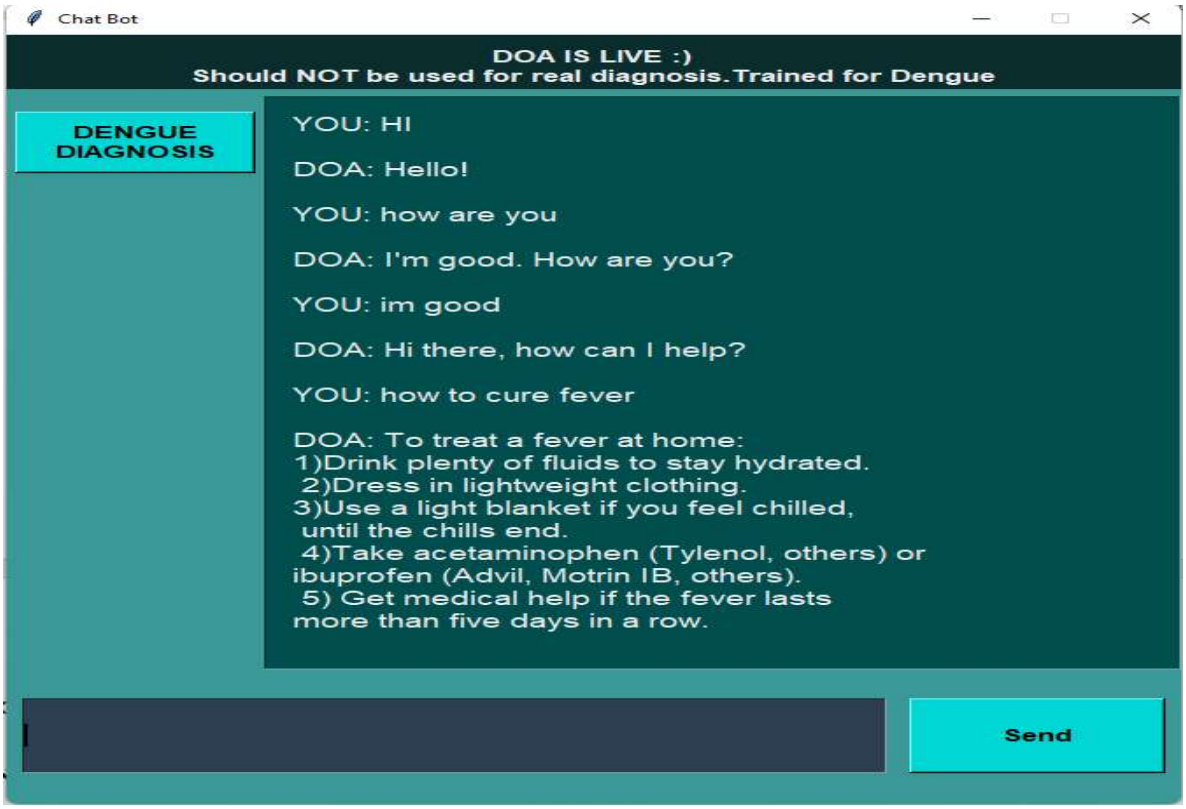
SYSTEM REQUIREMENTS



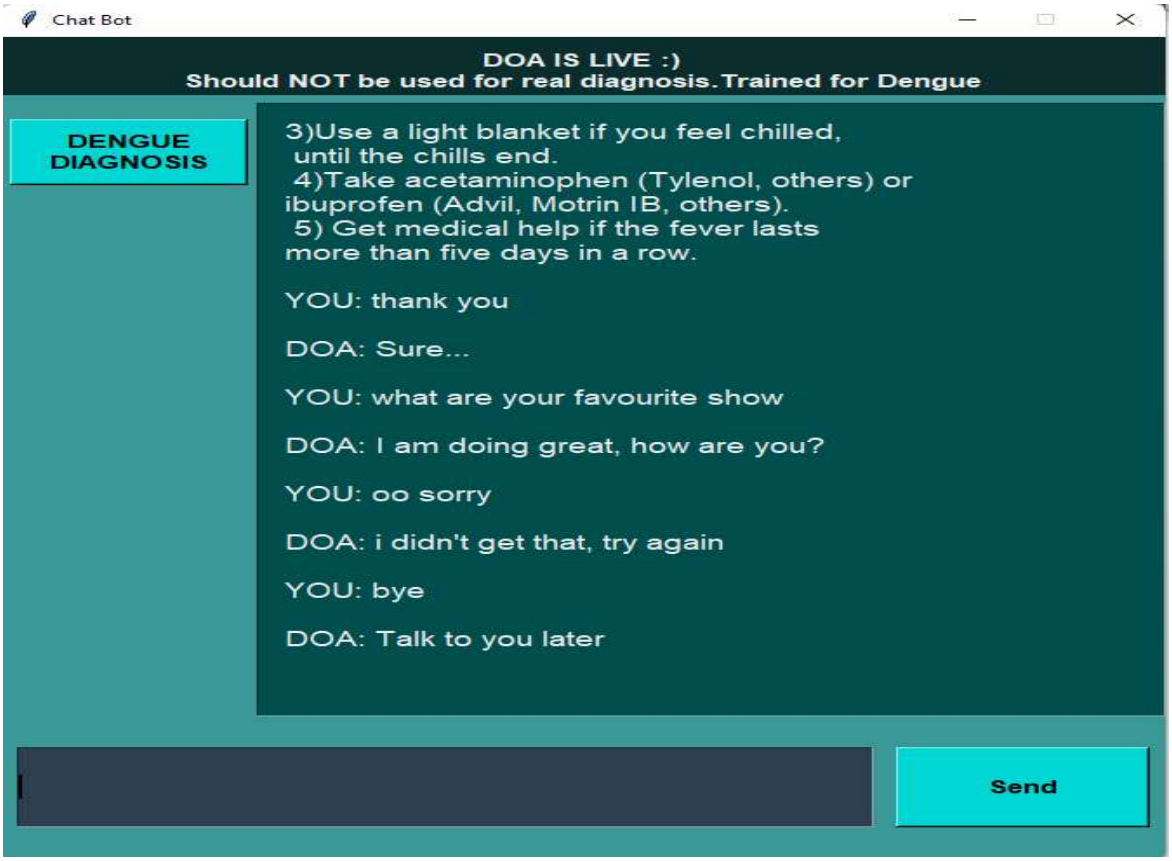
CHATBOT ACTIVATED



CHATTING WITH BOT



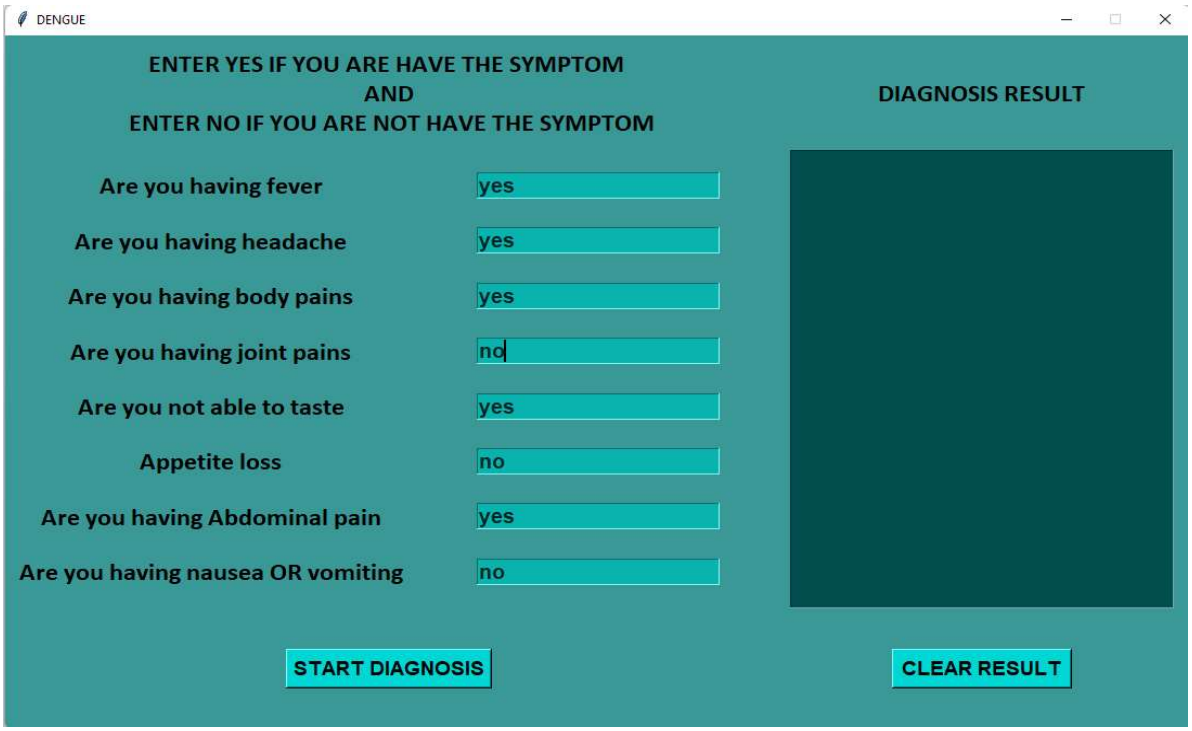
SCROLLING CHAT



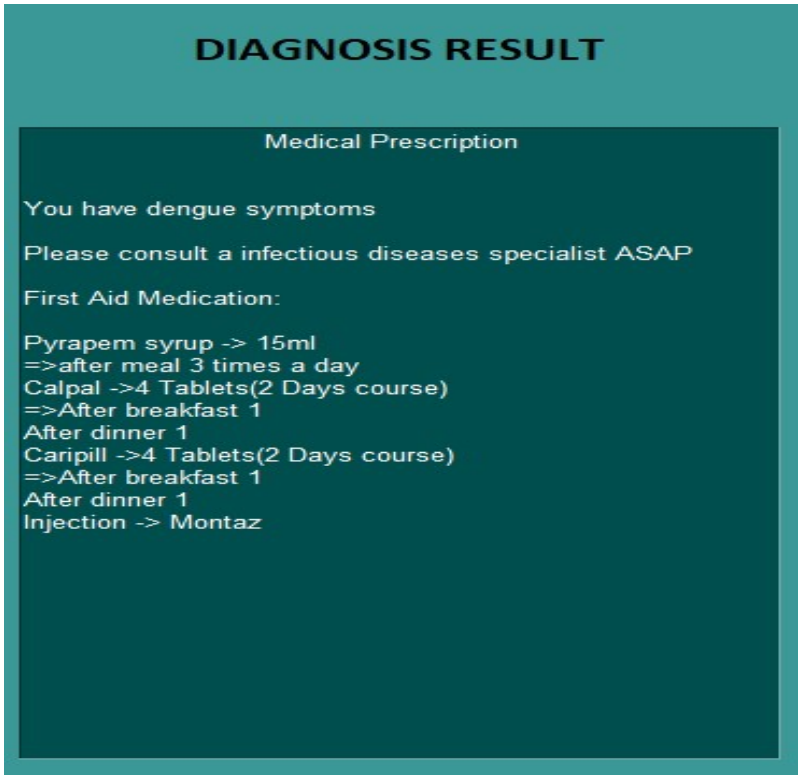
PREDICTION PAGE

USER INPUT PAGE

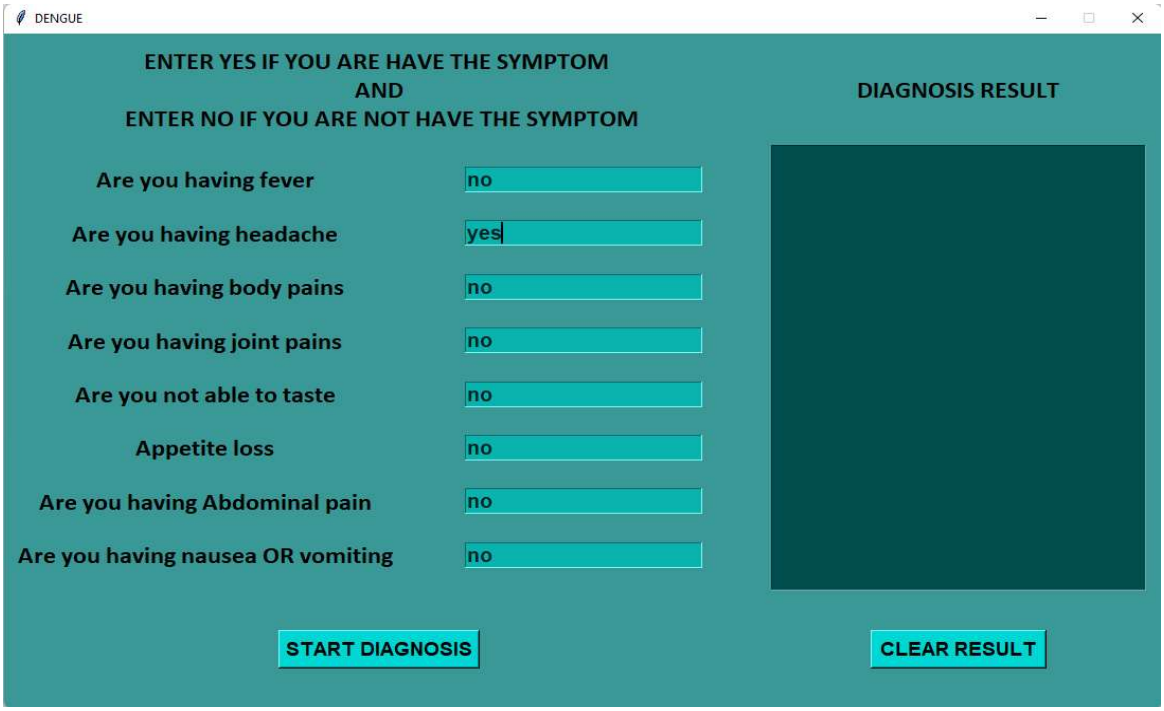
INPUT 1



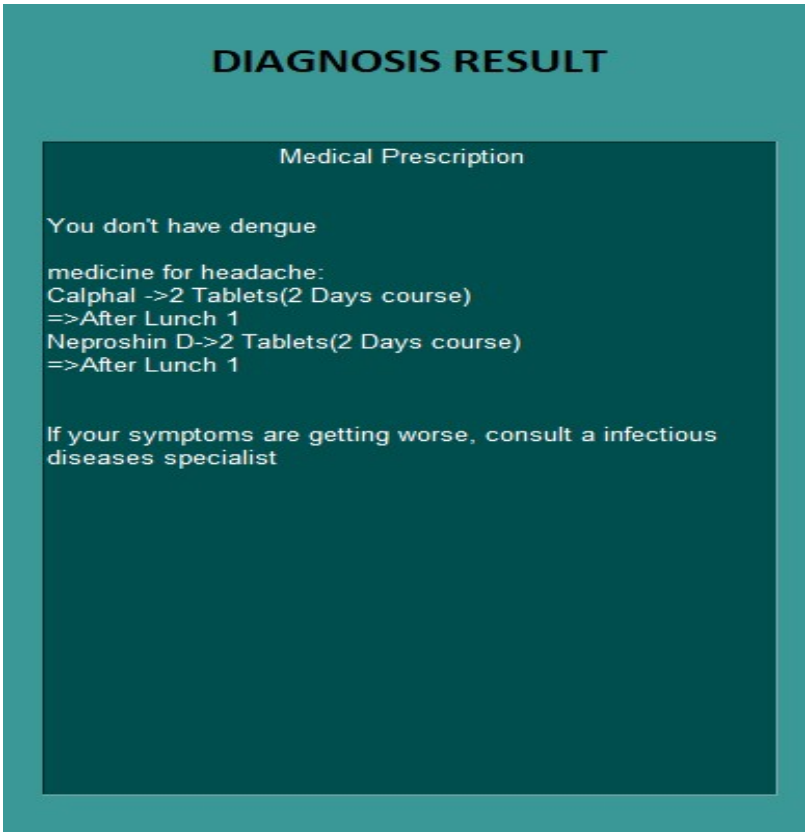
ANN RESULT 1



INPUT 2



ANN RESULT 2



CONCLUSION

CONCLUSION

This AI chatbot will reduce the burden of both medical staff and patients. As the world is Leaning on the online mode, it is very necessary to bring the same change in the medical field too. This project is a small step towards the goal to build an Indian AI chatbot for helping people with quick medical assistance.

Health check chatbot will reduce cost and man power requirements of medical industry and will be an efficient way to save time for the patients to get medical assistant.

REFERENCES

REFERENCES

[1] Khanna, A., Pandey, B., Vashishta, K., Kalia, K., Bhale, P., Das, T.: A study of today's A.I. through chatbots and rediscovery of machine intelligence. Int. J. u- e-Serv. Sci. Technol. **8**, 277–284 (2015). https://doi.org/10.14257/ijunesst.2015.8.7.28

[2] chatbot Definition of chatbot in English by Lexico Dictionaries. https://www.lexico.com/en/definition/chatbot

[3] Abu Shawar, B.A., Atwell, E.S.: Chatbots: are they really useful? J. Lang. Technol. Comput. Linguist. **22**, 29–49 (2007) Google Scholar

[4] Klopfenstein, L., Delpriori, S., Malatini, S., Bogliolo, A.: The rise of bots: a survey of conversational interfaces, patterns, and paradigms. In: Proceedings of the 2017 Conference on Designing Interactive Systems, pp. 555–565. Association for Computing Machinery (2017) Google Scholar

[5] Turing, A.M.: Computing machinery and intelligence. Mind **59**, 433–460 (1950). https://doi.org/10.1093/mind/LIX.236.433MathSciNetCrossRefGoogle Scholar

[6] Weizenbaum, J.: ELIZA—a computer program for the study of natural language communication between man and machine. Commun. ACM **9**, 36–45 (1966). https://doi.org/10.1145/365153.365168CrossRefGoogle Scholar

[7] Brandtzaeg, P.B., Følstad, A.: Why people use chatbots. In: Kompatsiaris, I., et al. (eds.) Internet Science, pp. 377–392. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70284-1_30CrossRefGoogle Scholar

[8] Colby, K.M., Weber, S., Hilf, F.D.: Artificial paranoia. Artif. Intell. **2**, 1–25 (1971). https://doi.org/10.1016/0004-3702(71)90002-6CrossRefGoogle Scholar

[9] Wallace, R.S.: The anatomy of A.L.I.C.E. In: Epstein, R., Roberts, G., Beber, G. (eds.) Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, pp. 181–210. Springer, Cham (2009). https://doi.org/10.1007/978-1-4020-6710-5_13CrossRefGoogle Scholar

[10] Marietto, M., et al.: Artificial intelligence markup language: a brief tutorial. Int. J. Comput. Sci. Eng. Surv. **4** (2013). https://doi.org/10.5121/ijcses.2013.4301

[11] Molnár, G., Zoltán, S.: The role of chatbots in formal education. Presented at the 15 September 2018Google Scholar

[12] Siri. https://www.apple.com/siri/

[13] Personal Digital Assistant - Cortana Home Assistant – Microsoft. https://www.microsoft.com/en-us/cortana

[14] What exactly is Alexa? Where does she come from? And how does she work? https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/

[15] Google Assistant, your own personal Google. https://assistant.google.com/

[16] IBM Watson. https://www.ibm.com/watson

[17] Scopus - Document search. https://www.scopus.com/search/form.uri?display=basic

[18] Colace, F., De Santo, M., Lombardi, M., Pascale, F., Pietrosanto, A., Lemma, S.: Chatbot for e-learning: a case of study. Int. J. Mech. Eng. Robot. Res. **7**, 528–533 (2018). https://doi.org/10.18178/ijmerr.7.5.528-533CrossRefGoogle Scholar

[19] Ranoliya, B.R., Raghuwanshi, N., Singh, S.: Chatbot for university related FAQs. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, pp. 1525–1530 (2017) Google Scholar

BIBLIOGRAPHY

1) Make Your Own Neural Network is a step-by-step journey through the mathematics of neural networks and making your own using the Python computer language written by Tariq Rashid.

2) Artificial Intelligence: A Modern Approach gives you detailed information about the changes that have taken place in the field of artificial intelligence written by Stuart Russell.

3) Deep Learning Illustrated This book talks about many powerful new artificial intelligence capabilities and algorithm performance written by Jon Krohn with Grant Beyleveld, and Aglae Basens.

4) TensorFlow in 1 Day: Make your own Neural Network It has a most authentic graph computations feature which helps you to visualize and designed neural network. This useful Machine learning book offers both convolutions as well as Recurrent Neural network written by Krishna Rungta.

5) Python Machine Learning, 1st Edition It helps you to learn the best practices and methods to improve and optimize machine learning systems and algorithms written by Sebastian Raschka.

APPENDIX

11. APPENDIX

11.1 List of Tables

11.2 List of Figures