# WEB BASED CHAT BOT

**A MAJOR PROJECT REPORT SUBMITTED**

**TO**

**OSMANIA UNIVERSITY, HYDERABAD**

**IN PARTIAL FULFILMENT OF THE AWARD FOR THE DEGREE OF**
**MASTER OF COMPUTER APPLICATIONS**

**Submitted by**

**TANNEERU SAI KUMAR**

**1325-19-862-069**

**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS**
**AURORA'S POST GRADUATE COLLEGE (MCA)**
**(Affiliated to Osmania University)**
**Ramanthapur, Hyderabad – 500013**

**AURORA'S POST-GRADUATE COLLEGE (MCA)**
**Ramanthapur, Hyderabad – 500 013**

# CERTIFICATE

This is to certify that the project work entitled

**WEB BASED CHAT BOT**

Is a bonafide work done by

**TANNEERU SAI KUMAR**
**1325-19-862-069**

as part of the curriculum in the

**DEPARTMENT OF COMPUTER SCIENCE**
**AURORA'S POST GRADUATE COLLEGE (MCA)**
**Ramanthapur, Hyderabad – 500013**

In partial fulfillment of requirement for award of

Master of Computer Applications

Osmania University, Hyderabad

This work has been carried out under our guidance.

Internal Guide Head of Department Principal

Internal Examiner External Examiner

IntelliCloud Apps

**06/07/2022**

# CERTIFICATE

This is to certify that **Mr. TANNEERU SAI KUMAR**, studying in **MCA** at **Aurora's Pg College(MCA)** with enrolment no **1325-19-862-069** successfully completed project work titled **"Web Based Chat bot"** in Python technology as part of internship.

He has done this project using **Python** platform during the period from **05/04/2022** to **06/07/2022** under the guidance and supervision of **Mr. Alluri Manda, Sr. Software Developer, IntelliCloud Apps Pvt Ltd,** Hyderabad**.**

He has completed the assigned project well within the time frame and is sincere, hard working and his conduct during his project is commendable.

We wish all the best to his future endeavors

**For IntelliCloud Apps Private Limited**

INTELLICLOUD APPS PVT. LTD.
HYDERABAD

[illegible]

# DECLARATION

I **TANNEERU SAI KUMAR** hereby declare that the project entitled **“WEB BASED CHAT BOT”** has been carried out by me in the **IntelliCloud Apps Pvt Ltd.** This project is submitted to Osmania University Hyderabad in partial fulfillment of requirements for the award of the degree of “MASTER’S OF COMPUTER APPLICATIONS”. The results embodied in this dissertation have not been submitted to any other University or institution for the award of Degree or Diploma.

(Signature of the student)

**TANNEERU SAI KUMAR**

1325-19-862-069

# ACKNOWLEDGEMENT

I would like to express deep gratitude and respect to all those people behind the scene who guided, inspired in the completion of this project work.

I wish to convey my sincere thanks to Principal and Head of Department of Computer Science, our project in charge **Mr. MOHAMMED ISMAIL** and project guide **Mr. Alluri Manda ,** for giving me the required guidance during this project work.

Last but not least I am very thankful to the faculty members of my college and friends for their suggestions and help me successfully completing this project.

**TANNEERU SAI KUMAR**

1325-19-862-069

# INDEX

| TITLE | PAGE NO |
|---|---|
| 1. Organization Profile | 07 |
| 2. Abstract | 09 |
| 3. System Analysis | 19 |
| 3.1 Description of Modules | |
| 3.2 Feasibility Study. | |
| 3.3 Software Requirement Specification | |
| 4.System Design | 27 |
| 4.1 UML Diagrams | |
| 5.Implementation | 34 |
| 5.1 Sample Source Code | |
| 5.2 Screen Shots | |
| 6.Testing | 55 |
| 7.Scope | 60 |
| 8.Conclusion | 61 |
| 9.Bibliography | 62 |
| 10.Webliography | 63 |

# 1. ORGANIZATION PROFILE

Founded in 2014, INTELLICLOUD APPS PVT. LTD. located at Kphb, Hyderabad, has a rich background in developing Mobile Applications and IOT projects, especially in solving latest problems, Software Development and continues its entire attention on achieving transcending excellence in the Development and Maintenance of Software Projects and Products in Many Areas.

In today's modern technological competitive environment, students in computer science stream want to ensure that they are getting guidance in an organization that can meet their professional needs. With our well-equipped team of solid information systems professionals, who study, design, develop, enhance, customize, implement, maintain, and support various aspects of information technology, students can be sure.

We understand the customer needs and develop their quality of professional life by simply making the technology readily usable for them. We practice exclusively in software development, network simulation, search engine optimization, customization, and system integration. Our project methodology includes techniques for initiating a project, developing the requirements, making clear assignments to the project team, developing a dynamic schedule, reporting status to executives and problem solving.

**About The People**:

As a team we have the clear vision and realize it too. As a statistical evaluation, the team has more than 40,000 hours of expertise in providing real-time solutions in the fields of Mobile Apps Development in Android, IOS, and Flutter, Web Designing, Web Development, Cloud Computing, Image Processing and Implementation, client Server Technologies in Java,(J2EE), ANDROID, Python, PHP, Oracle.

**Our Vision**:

“Impossible as Possible” this is our vision; we work according to our vision.

# 2. ABSTRACT

The days of solely engaging with a service through a keyboard are over. Users interact with systems more and more through voice assistants and chat bots. A chat bot is a computer program that can converse with humans using Artificial Intelligence in messaging platforms. Every time the chat bot gets input from the user, it saves input and response which helps chat bot with little initial knowledge to evolve using gathered responses. With increased responses, precision of the chat bot also gets increases. The ultimate goal of this project is to add a chat bot feature and API. This project will investigate how advancements in Artificial Intelligence and Machine Learning technology are being used to improve many services. Specifically it will look at development of chat bots as a channel for information distribution. The program selects the closest matching response from closest matching statement that matches input utilizing Word Net (NLTK), it then chooses response from known selection of statements for that response. This project aimed to implement online chat bot system to assist users who access websites, using tools that expose Artificial Intelligence methods such as Natural Language Processing, allowing users to communicate with web chat bot using natural language input and to train chat bot using appropriate Machine Learning methods so it will be able to generate a response. There are numerous applications that are incorporating a human appearance and intending to simulate human dialog, yet in most part of the cases knowledge of chat bot is stored in a database created by a human expert.

# INTRODUCTION

In this Python web-based project we are going to build a chat bot using machine learning and flask techniques. The chat bot will be trained on the dataset which contains categories (intents), pattern and responses. We use a special artificial neural network (ANN) to classify which category the user's message belongs to and then we will give a random response from the list of responses.

The improvements in the fields of inter-networking and information technology have been intricate in executing an Artificial Intelligent (AI) systems. These systems are drawing nearer of human activities, for example, choice emotionally supportive networks, robotics, natural language processing, and so forth. Indeed, even in the artificial intelligent fields, there are some hybrid strategies and adaptive techniques that make increasingly complex techniques. That, yet these days there are additionally several Natural Language Processing (NLP) and intelligent systems that could comprehend human language. Artificial intelligent systems learn themselves and retrieve insight by perusing required electronic articles that have been existed on the web.

A chat bot (otherwise called a chatterbox, Bot, or Artificial Conversational Entity) is an AI program that copies human discussions including content and communication in natural language utilizing artificial intelligence methods, for example, Natural Language Processing (NLP), picture and video processing and voice analysis. Chat bot for Websites has been created utilizing artificial intelligence algorithms that examine the user queries. This chat bot system is an internet application and website that gives an answer to the broken down queries of an user. Users simply need to choose the classification for inquiries and afterward ask the question to the bot that utilizes for noting it. Artificial intelligence has been incorporated to respond to the user's inquiries. Then the user can procure the fitting solutions to their inquiries.

The appropriate responses are given utilizing artificial intelligence algorithms. Users need to enlist to the system and needs to access websites. After opening the website there will be chat bot is there which users can chat by asking questions related with company activities. The system answers to users' queries with the assistance of effective Graphical User Interface (GUI). The user can question about the company related activities with the assistance of this web application. A chat bot is an Artificial Intelligence program that can converse with people in natural language, the manner in which we collaborate with one another. It can trade a human for some undertakings of replying inquiries.

A chat bot is a specialist that assists users in utilizing natural language. It was worked as an endeavor to trick people. A few uses of chat bots, for example, User care, customer support and so on utilizes Artificial Intelligence Markup Language (AIML) to visit with users. One of the foremost objectives of chat bots is to take after a smart human and entangle the recipient of the discussion to comprehend the genuine working along with different designs and abilities for their use has generally widened. These chat bots can demonstrate adequate to trick the user to believe that they are "talking" to an individual, however, they are limited in improving their insight base at runtime, and have typically next to zero methods for keeping track of all the discussion information. Chat bots utilize AI to arrive at counterfeit intelligence helping them to comprehend the user question, what's more, give a suitable reaction. The chat bots are created utilizing the Artificial Intelligence Markup Language (AIML) for imparting or cooperating with the user. This comprises software that will be made up of utilizing Artificial Intelligence and will assist the user in chatting with a machine. The user can ask the systems like typically did to other humans.

## EXISTING SYSTEM

Changing of requirements As an industrial project to build a product, we must follow the requirement from the user. However, because the project's goal is to be used by the business team, but it is responsible by the technical team, the requirement changed a lot in the middle after a meeting with the business team. The business team want a simple bot that can give recommendation immediately. We had to archive what we had done before and build a new one.

**Lacking of training data:** The quantity and quality of training data is critical to the performance to a machine learning model. However, because some confidential and privacy reasons, the business team cannot provide enough data for us, and we had to make up data by our own. For the machine learning model, we generate some fake data based on our daily life experience, which is really based, although with a good accuracy on the fake data.

**Unstable API version:** Because API service we are using are still under development, and we cannot fix to a version for the API, the API may changes overtime. Moreover, there are inconsistencies between the APIs and their documents or sample codes.

**Not familiar with the PHP language and .NET framework:** None of the three of us has previous knowledge with PHP language. Programming in a new language in such a huge framework is quite challenging for us at the beginning of the project. However, when we comes to the later phases, we are more used to that.

# PROPOSED SYSTEM

**Importing and loading the data file**

First, make a file name as trainning.py. We import the necessary packages for our chat bot and initialize the variables we will use in our Python project.

The data file is in JSON format so we used the json package to parse the JSON file into Python (data .Json).

**Preprocess data**

When working with text data, we need to perform various preprocessing on the data before design an ANN model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list. We also create a list of classes for our tags.

Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

**Creating training and testing data**

Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn’t understand text so we will convert text into numbers.

## Build the model

We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as 'model.h5' (training.py).

## Predict the response (Flask web-based GUI)

Now to predict the sentences and get a response from the user to let us create a new file 'app.py'using flask web-based framework.

**Note :** for making flask app we need to make to folders name as static and templates and app.py files.

- static folder contains a subfolder with name styles. The styles folder contain css file with the name style.css
- Templates folder HTML file with the name of index.html
- app.py file for run the flask app using IDE.

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve a random response from the list of responses.

Again we import the necessary packages and load the 'texts.pkl' and 'labels.pkl' pickle files which we have created when we trained our model:

To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class. After predicting the class, we will get a random response from the list of intents.

# ADVANTAGES & DISADVANTAGES

## ADVANTAGES

**Accessible anytime:** I'm sure most of you are always kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls. Since chat bots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break. This improves your customer UX and helps you rank highly in your sector. Another advantage of this instant response is that you can also skillfully craft your chat bot to maintain your image and brand.

**Handling Capacity:** Unlike humans who can only communicate with one human at a time, chat bots can simultaneously have conversations with thousands of people. No matter what time of the day it is or how many people are contacting you, every single one of them will be answered immediately. Imagine you own a restaurant, and you have a good reputation for your food of which most of your revenues come from delivery. As the demand keeps rising, you will have more customers to take orders from but very few staff to attend them all. Having a chat bot would eliminate such problem and cater to each and every person and ensure that no order is missed. Companies like Taco Bell and Dominos are already using chat bots to arrange delivery of parcels.

**Flexible attribute:** Chat bots have the benefit that it can quite easily be used in any industry. Unlike other products where you have to do a lot of development and testing to change platforms, chat bots are relatively easy to switch. One has to just train the bot by giving the right conversation structure and flow to switch its current field or industry. Or if there is a lot of back and forth between two sections of the industry say customer support and sales, then you could have custom built presets which would already have the conversation flow and structure to carry out the interactions with the user.

**Customer Satisfaction:** Humans are bound to change of emotions. Chat bots, on the other hand, are bound by some rules and obey them as long as they're programmed to. They will always treat a customer in the perfect way no matter how rough the person is or how foul language the person uses. Not everyone orders the same food everyday , people's choices may change everyday . In this case, it can use your order history to make suggestions for the next order, learn your address details and much more. Customers love this smooth interaction and want all their transactions to be as simple as possible.

**Cost Effective:** Hiring a human for a job is never a cheap affair, and it will be expensive if your revenue are not high or sales targets are not met and would create havoc in the business. Due to the boundaries of human beings, a single human can only handle one or two people at the same time. More than that would be extremely tough for the employee. Chat bots could help solve this age-old problem. As one chat bot is equal to loads of employees, it can easily communicate with thousands of customers at the same time. We would only need a handful of people to jump into conversations sometimes when necessary. Hence, it would drastically bring down the expenses and bring about a steep rise in revenue and customer satisfaction.

**Faster On boarding:** Before you want to accomplish a task, you first must learn how to work on the task and complete it. Only then will they be considered fit for the job. There is a continuous teaching involved in every level of hierarchy the employee will go through. Also, there will be a lot of change in the employees, some stay, some get fired, some more join in etc. What we want to say is, employees will change, it's a fact. And this would require you to allot a lot time of your employees into grooming the new joinees. Chat bots could eliminate that time to almost zero, but provide a very clean and easy to understand conversation flow and structure that needs to be maintained by the chat bot. No doubt there will be changes in this too, but it will rather take a fraction of your time to resolve as compared to human employees.

## Disadvantages

**Too many functions:** Most of developers strive to create a universal chat bot that will become a fully-fledged assistant to user. But in practice functional bots turn out not to cope with the majority of queries. They often do not understand the user, forget what they were told 5 minutes earlier, and have many other disadvantages. And that is no wonder, as the development of a universal bot, which would understand natural language and could evaluate context, takes years of hard work for a team of experienced programmers. And even in this case, such programs should be constantly improved while in service. However, modern technologies allow building rather useful bots to perform specific actions such as booking train tickets or providing support to bank customers.

**Primitive algorithms:** There are two types of bots: based on artificial intelligence, being able to learn in the process of communication programmed for specific behavior scenarios. Artificial intelligence chat bots are considered to be better, as they can respond depending on the situation and context. But the development of complex algorithms is required for this purpose. Meanwhile, only IT giants and few developers possess such powerful technological base. Therefore, it would be better for ordinary companies to focus on the second variant of bots, as they are more reliable and simpler. Namely for the reason they do not possess intelligence, they will not be able to adopt rude communication patterns and get beyond the control of creators.

**Complex interface:** Talking to a bot implies talking in a chat, meaning that a user will have to write a lot. And in case a bot cannot understand the user's request, he will have to write even more. It takes time to find out which commands a bot can respond to correctly, and which questions are better to avoid. Thus, talking to a chat bot does not save time in the majority of cases. Perhaps the efficiency of virtual assistants will increase due to the implementation of voice recognition function in the future. But for the time being their functional capabilities are very restricted, and they can be truly useful only in a few business areas.

# HARDWARE AND SOFTWARE REQUIREMENTS

## HARDWARE REQUIREMENTS:

| | | |
|---|---|---|
| • Ram | : | 8GB |
| • Processor | : | Intel core i5 |
| • Speed | : | 2.4GHz |

• Active internet connection

## SOFTWARE REQUIREMENTS:

| | | |
|---|---|---|
| • Operating System | : | Windows 11/Windows 10 |
| • Language | : | Python 3.8, Anaconda3(64bit) |
| • Software | : | Spyder Notebook/Jypter Notebook |
| • Modules | : | Nltk, Numpy, Keras, Pickle, Json , Html, Css |
| • Server (Back-end) | : | Flask |

# 3. SYSTEM ANALYSIS

Analysis to products is already a common sense among web and app developers for long, and sure the same applies to chat bots too. Chat bot analysis is more complicated than web and app analysis. Since the interaction between user and chat bot includes three major parts: **user interface**, **logic** and **conversation**, page view and message count are not the only things you should worry about anymore.

For a service as complicated as chat bot, what traditional analytics tools are offering is way far from enough. New, specialized analytics that measures user engagement and conversational sentiments are needed.

To me, a good chat bot analytics tool should monitor all conversations between the bot and its users so the bot owner knows the entire customer journey, and when human agents should join conversations if needed. One can then take different actions towards each segmented customers and leverage on data to help increase user engagement.

## 3.1. DESCRIPTION OF MODULES

**Bots:**

An Internet bot is an automated software application. It can run any range of tasks and does so repetitively. The implementation of bots on the Internet is so widespread that bots made up 50% of all online traffic in 2016 . Some of the tasks that bots perform are feed fetchers, commercial crawlers, monitoring, and search engine bots. For example, feed fetchers change the display of websites when they are accessed for mobile users and search engine bots collect metadata that allows the search engine to perform. These tasks shape the Internet as we see it daily.

**Tensorflow:**

TensorFlow enables you to build dataflow graphs and structures to define how data moves through a graph by taking inputs as a multi-dimensional array called Tensor. It allows you to construct a flowchart of operations that can be performed on these inputs, which goes at one end and comes at the other end as output.

Tensorflow architecture works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

**Keras:**

Keras is an API designed for human beings**,** not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages.

**Pickle:**

It convert Python objects to streams of bytes and back.

Texts.pkl – This is a pickle file in which we store the words Python object using Nltk that contains a list of our vocabulary.

Labels.pkl – The classes pickle file contains the list of categories(Labels).

**Nltk:**

The chat bot uses the Natural Language Processing Toolkit(NLTK) to process the textual information.

Natural Language Toolkit used to deal with NLP applications and the chat bot is one of them. We will now advance our rule-based chat bots using the NLTK.

**Flask:**

The Flask is a Python micro-framework used to create small web applications and websites using Python. Flask works on a popular templating engine called Jinja2, a web templating system combined with data sources to the dynamic web pages.

Now start developing the Flask framework based on the above Chatter Bot.

We have already installed the Flask in the system, so we will import the Python methods we require to run the Flask micro server.

app.py – This is the flask Python script in which we implemented web-based GUI for our chat bot. Users can easily interact with the bot.

## 3.2. FEASIBILITY STUDY

**Operational Feasibility**

All projects are feasible if they have unlimited resources and infinite time. But the development of software is plagued by the scarcity of resources and difficult delivery rates. It is necessary and prudent to evaluate the feasibility of a project at the earliest possible time. The three considerations are involved in the feasibility analysis

**Economical Feasibility**

This procedure is to determine the benefits and savings that are expected from a candidate system and compare with cost. If benefits outweigh cost then the decision is made to design and implement the system.

Otherwise further justification or alterations in proposed systems that have to be made if it is having a change of being approved. This is an ongoing effort that improves any feasibility costs spent on this project because here I am using open source environments.

**Technical Feasibility**

Technical feasibility centers on the existing operating system (hardware, software…etc) and to what extent it can support the proposed addition if the budget is a serious constraint. The technical feasibilities are important role in my project because here I am using the operating system.

## 3.3. SOFTWARE REQUIREMENT SPECIFICATION

**Introduction:**

The success of a software system can be accessed on the basis of its fulfillment of two interdependent types of requirements. First and foremost, the client focused requirements that focus on the functionalities given to the users of the system. Second, the context focused requirements which can be system, procedure and human necessities. The later type of requirements is suggested as non-functional requirements (Nfrs). Nfrs are systems goals that may impact the operational environment and design decisions an employer may look for the progression of the software. Besides other things Nfrs incorporate operational environment: hardware and software interfaces, accuracy, performance etc. Functional requirements portray the noticeable external input and output interactions with the system under scrutiny, despite the fact that non-functional requirements are those that drive unique conditions and qualities on the system to develop. Thus, system acceptance testing is centered on both functional and non-functional system's requirements.

**Introduction to Python:**

Python is a popular programming language. First developed in the late 80's by Guido van Rossum, Python is currently in its third version, released in 2008, although the second version originally released in 2000 is still in common usage.

**It is used for:**

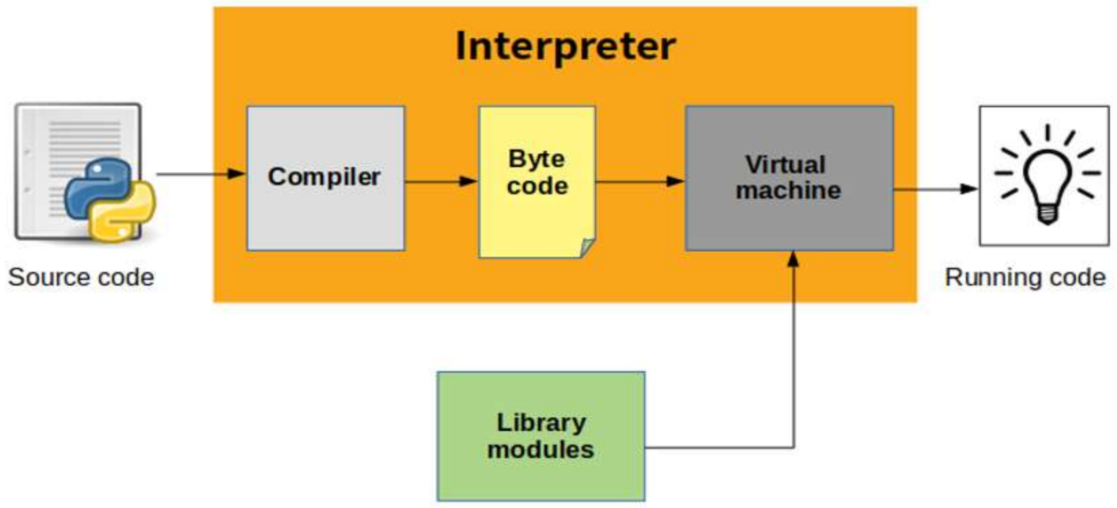
- web development (server-side)
- software development
- mathematics, system scripting

**What can Python do?**

- Python can be used on a server side to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping or for production-ready software development.
- The Python language itself is managed by Python Software Foundation, who offer a reference implementation of python called C Python under an open source license.

**Why Python?**

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.



**Compilation of code:**

Python first compiles your source code (.py file) into a format known as byte code. Compilation is simply a translation step and byte code is a lower-level, and platform-independent, representation of your source code. Compiled code is usually stored in.pyc files and is regenerated when the source is updated or when otherwise necessary. To distribute a program to people who already have Python installed, you can ship either the .py files or the .pyc files.

The byte code (.pyc file) is loaded into the Python runtime and interpreted by a Python Virtual Machine, which is a piece of code that reads each instruction in the byte code and executes whatever operation is indicated. Byte code compilation is automatic and the PVM is just part of the Python system that you have installed on your machine. The PVM is always present as part of the Python system and is the component that truly runs your scripts.

Technically, it's just the last step of what is called the Python interpreter. And this is how the process is done (very general). Of course, there are optimizations and caches to improve the performance.

Each time an interpreted program is run, the interpreter must convert source code into machine code and pull in the runtime libraries. This conversion process makes the program run slower than a comparable program written in a compiled language. Python do something clever to improve its performance. It compiles to byte code (.pyc files) the first time it executes a file. This improves substantially the execution of the code next time the module is imported or executed.

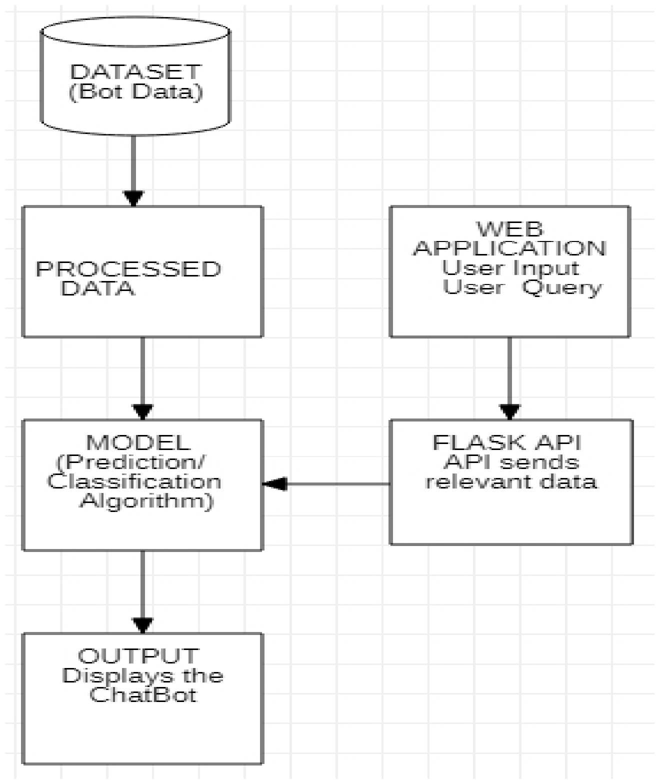
- Easy-to-learn − Python has few keywords, simple structure and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintaining.

# 4. SYSTEM DESIGN

## 4.1. UML DIAGRAMS:

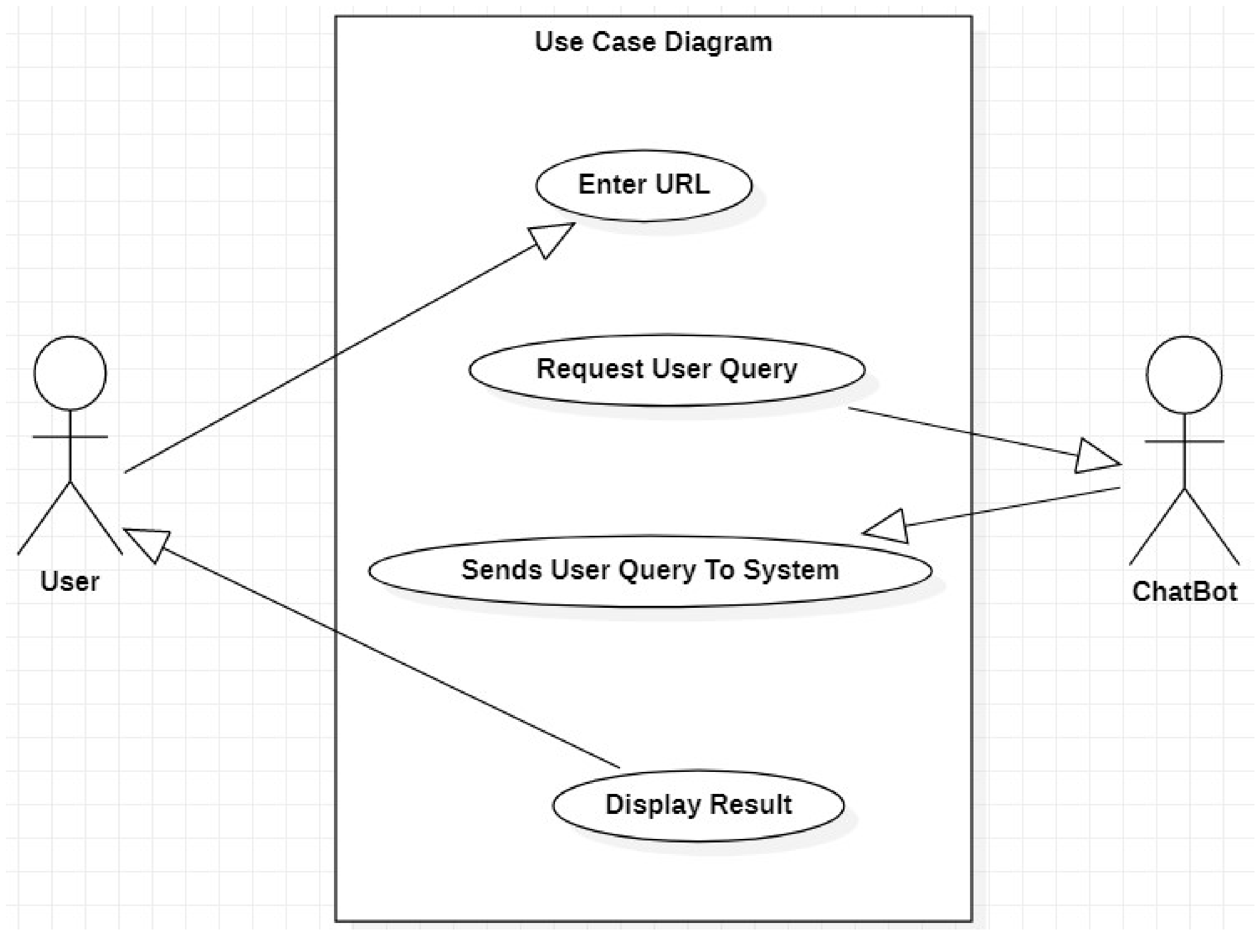
**Block Diagram:**

A block diagram is a specialized, high- level flowchart used in engineering. It is used to design new systems or to describe and improve existing ones. Its structure provides a high-level overview of major system components, key process participants, and important working relationships.



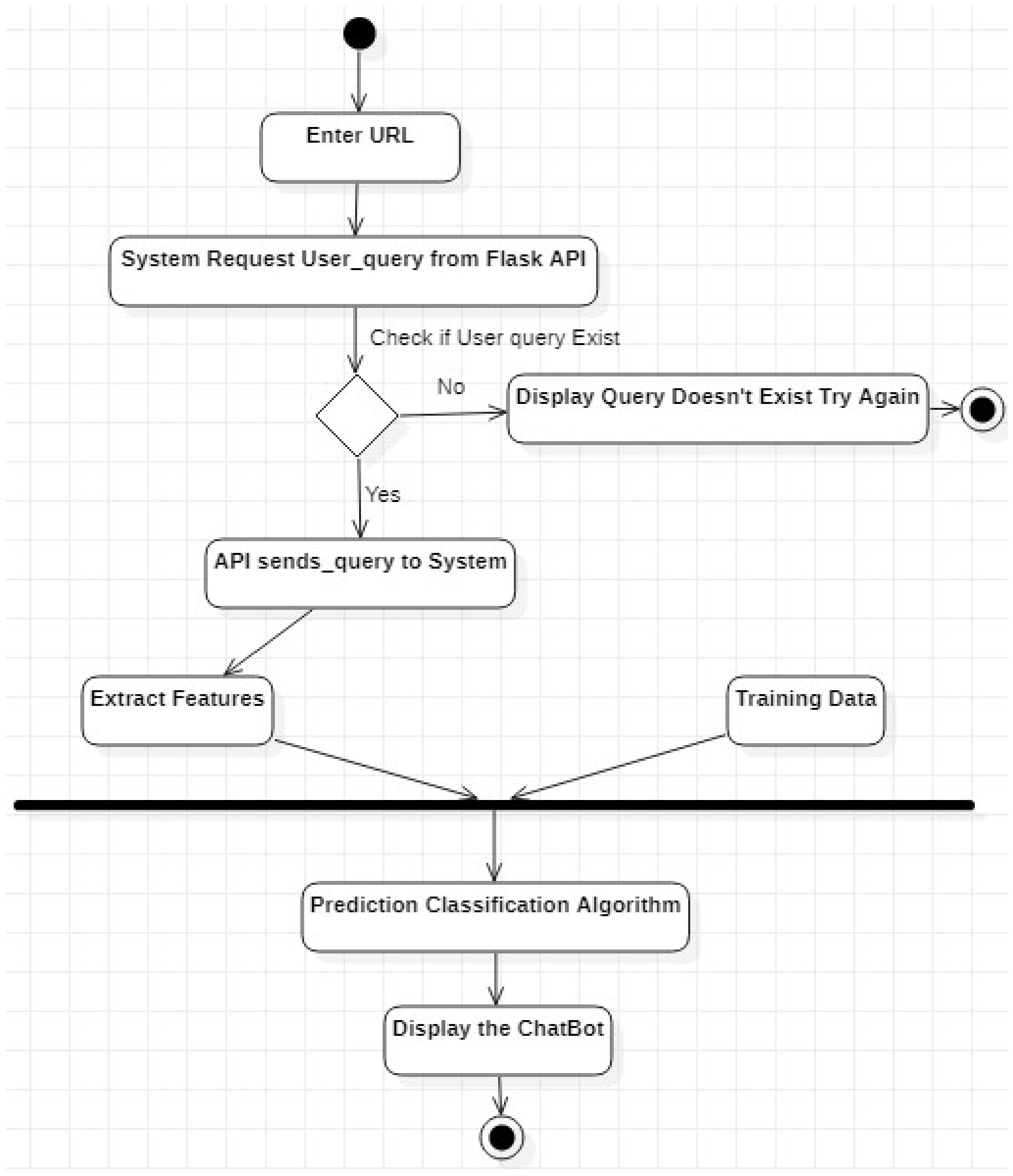
**Use case Diagram:**

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.



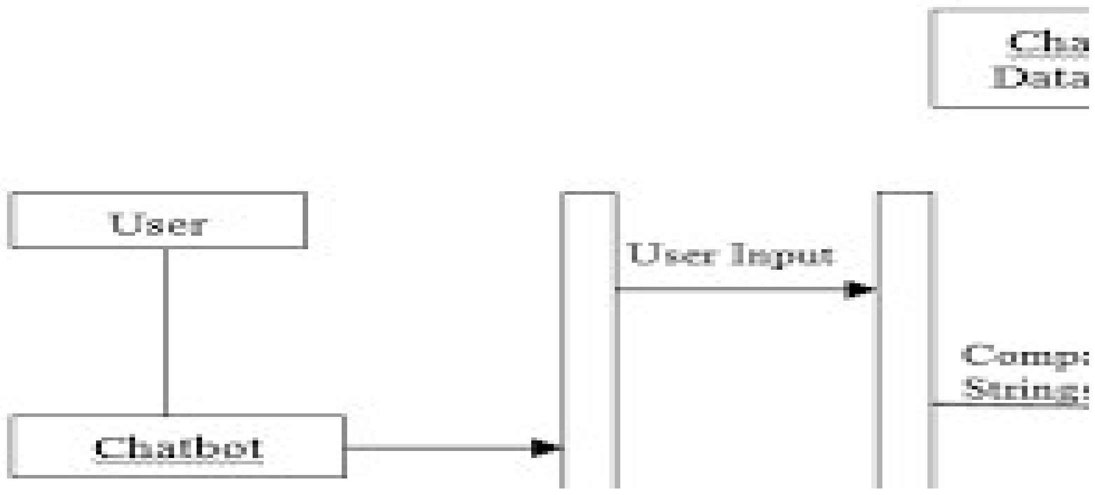
**Activity Diagram :**

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.


Enter URL
System Request User_query from Flask API
Check if User query Exist
No
Display Query Doesn't Exist Try Again
Yes
API sends_query to System
Extract Features
Training Data
Prediction Classification Algorithm
Display the ChatBot

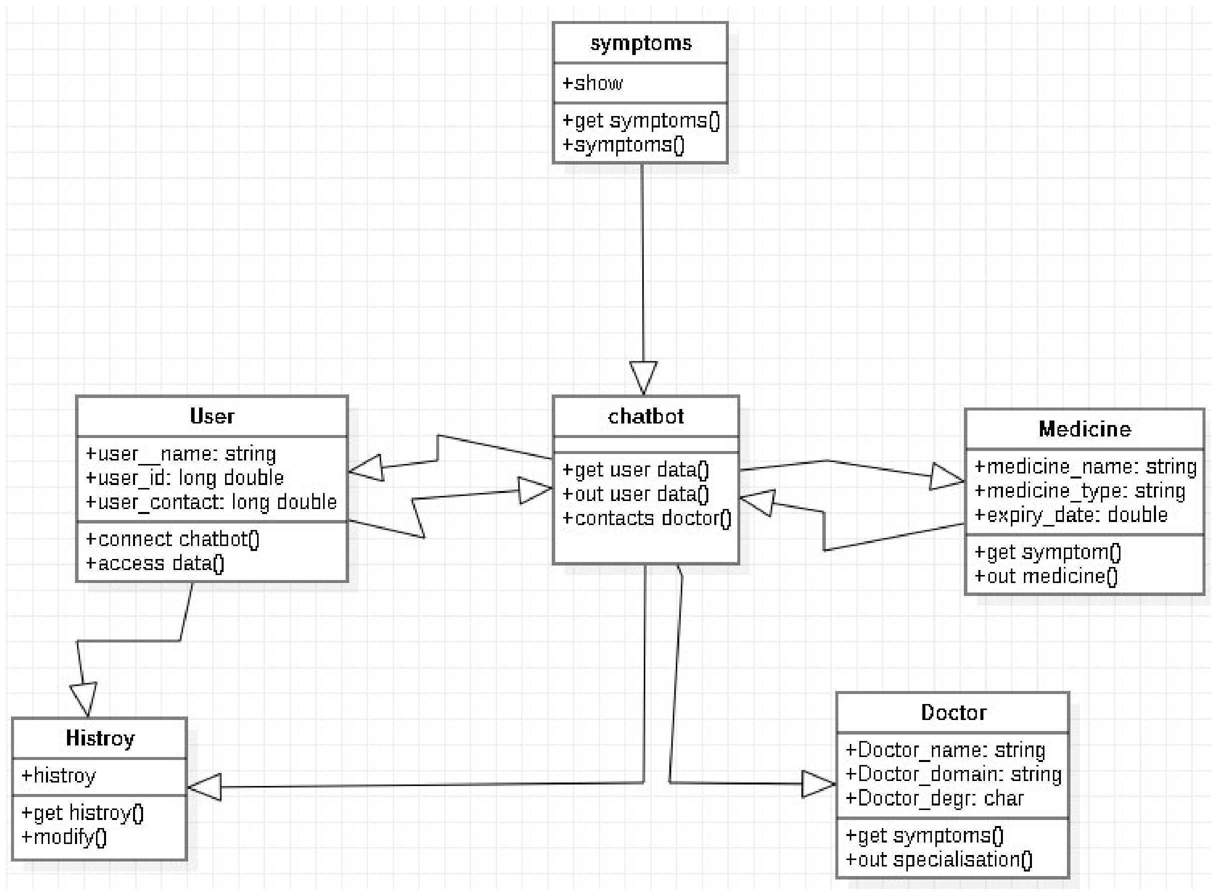
**SEQUENCE DIAGRAM:**

UML Sequence diagrams are a simple and instutive way of presenting the environment to explain a system's behavior. A map of the series displays an interaction structured in a time sequence.



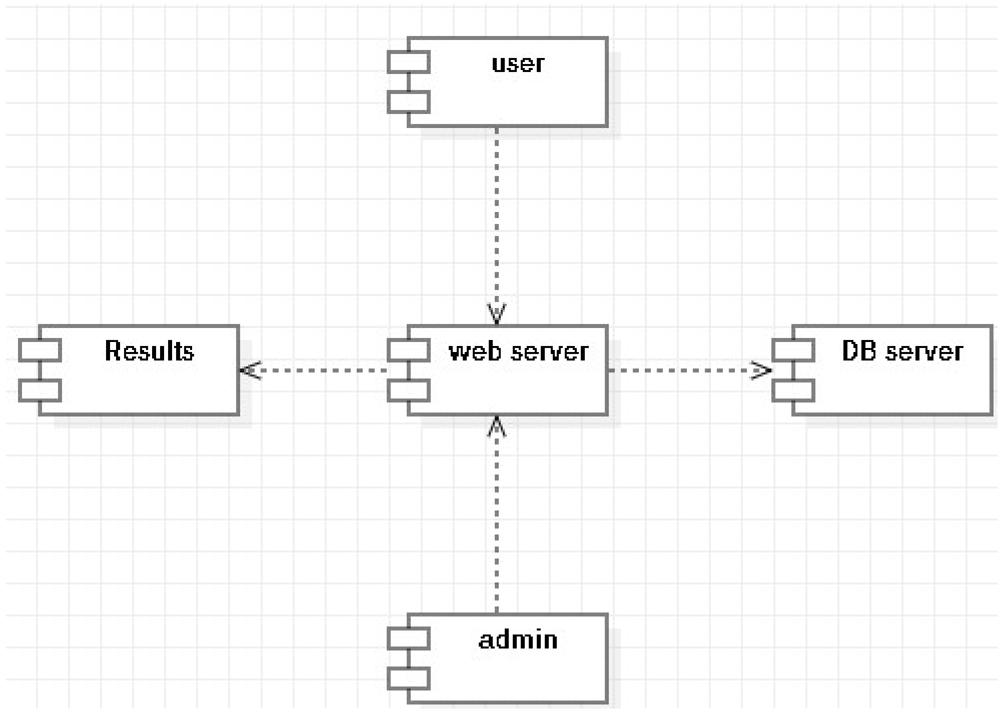
## CLASS DIAGRAM:

Class diagram describes the attributes and operations of a class and the constraints imposed on the system.



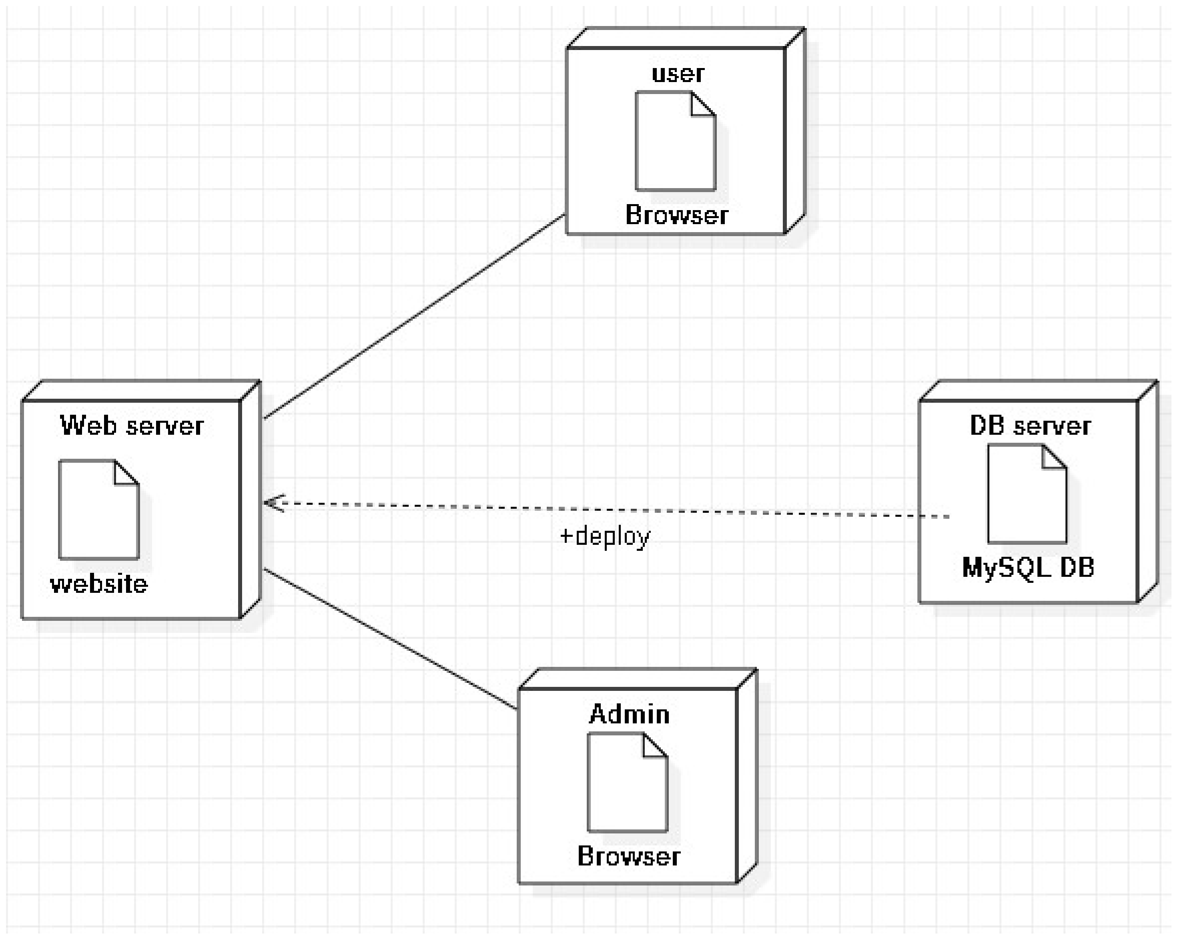
## COMPONENT DIAGRAM:

A component diagram, also known as a UML component diagram, **describes the organization and wiring of the physical components in a system**. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



## DEPLOYMENT DIAGRAM:

A deployment diagram is **a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them**. Deployment diagrams are typically used to visualize the physical hardware and software of a system



# 5. IMPLEMENTATION

This gives a brief description about implementation details of the system in terms of algorithms. Four algorithms are implemented namely Decision Tree, Multinomial Naïve Bayes, Random Forest and Bag of Words.

## 5.1. SAMPLE SOURCE CODE

**#install Libraries**

```
Pip install tensorflow
Pip install keras
Pip install pickle
Pip install nltk
Pip install flask
```

**#data.Json(file name)**

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola",
"Hello", "Good day"],
         "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there,
how can I help?"],
         "context": [""]
        },
        {"tag": "goodbye",
         "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye",
"Till next time"],
         "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
         "context": [""]
        },
        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks",
"Thanks for helping me"],
         "responses": ["Happy to help!", "Any time!", "My pleasure"],
         "context": [""]
        },
        {"tag": "noanswer",
         "patterns": [],
         "responses": ["Sorry, can't understand you", "Please give me more info",
"Not sure I understand"],
         "context": [""]
        },
        {"tag": "options",
         "patterns": ["How you could help me?", "What you can do?", "What help you
provide?", "How you can be helpful?", "What support is offered"],
         "responses": ["I can guide you through Adverse drug reaction list, Blood
pressure tracking, Hospitals and Pharmacies", "Offering support for Adverse drug
reaction, Blood pressure, Hospitals and Pharmacies"],
         "context": [""]
        },
        {"tag": "adverse_drug",
         "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a list of drugs causing adverse behavior", "List all drugs suitable for patient with adverse reaction", "Which drugs dont have adverse reaction?" ],
         "responses": ["Navigating to Adverse drug reaction module"],
         "context": [""]
        },
        {"tag": "blood_pressure",
         "patterns": ["Open blood pressure module", "Task related to blood pressure", "Blood pressure data entry", "I want to log blood pressure results", "Blood pressure data management" ],
         "responses": ["Navigating to Blood Pressure module"],
         "context": [""]
        },
        {"tag": "blood_pressure_search",
         "patterns": ["I want to search for blood pressure result history", "Blood pressure for patient", "Load patient blood pressure result", "Show blood pressure results for patient", "Find blood pressure results by ID" ],
         "responses": ["Please provide Patient ID", "Patient ID?"],
         "context": ["search_blood_pressure_by_patient_id"]
        },
        {"tag": "search_blood_pressure_by_patient_id",
         "patterns": [],
         "responses": ["Loading Blood pressure result for Patient"],
         "context": [""]
        },
        {"tag": "pharmacy_search",
         "patterns": ["Find me a pharmacy", "Find pharmacy", "List of pharmacies
nearby", "Locate pharmacy", "Search pharmacy" ],
         "responses": ["Please provide pharmacy name"],
         "context": ["search_pharmacy_by_name"]
        },
        {"tag": "search_pharmacy_by_name",
         "patterns": [],
         "responses": ["Loading pharmacy details"],
         "context": [""]
        },
        {"tag": "hospital_search",
         "patterns": ["Lookup for hospital", "Searching for hospital to transfer
patient", "I want to search hospital data", "Hospital lookup for patient", "Looking
up hospital details" ],
         "responses": ["Please provide hospital name or location"],
         "context": ["search_hospital_by_params"]
        },
        {"tag": "search_hospital_by_params",
         "patterns": [],
         "responses": ["Please provide hospital type"],
         "context": ["search_hospital_by_type"]
        },
        {"tag": "search_hospital_by_type",
         "patterns": [],
         "responses": ["Loading hospital details"],
         "context": [""]
        }
```

```
   ]
}
```

**#training.py(file name) #Python Code**

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('data.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)

        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates

words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
ignore_words]
words = sorted(list(set(words)))

# sort classes
classes = sorted(list(set(classes)))

# documents = combination between patterns and intents
print (len(documents), "documents")

# classes = intents
print (len(classes), "classes", classes)
```

```
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)



pickle.dump(words,open('texts.pkl','wb'))
pickle.dump(classes,open('labels.pkl','wb'))

# create our training data
training = []

# create an empty array for our output
output_empty = [0] * len(classes)

# training set, bag of words for each sentence
for doc in documents:

    # initialize our bag of words
    bag = []

    # list of tokenized words for the pattern
    pattern_words = doc[0]

    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in
pattern_words]
```

```
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])

# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)

# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")


# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd
output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient
gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
verbose=1)
model.save('model.h5', hist)

print("model created")
```

## #Static/Styles/Style.css (file name) #CSS Code

```
:root {
  --body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
  --msger-bg: #fff;
  --border: 2px solid #ddd;
  --left-msg-bg: #ececec;
  --right-msg-bg: #579ffb;
}

html {
  box-sizing: border-box;
}

*,
*:before,
*:after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-image: var(--body-bg);
  font-family: Helvetica, sans-serif;
}

.msger {
  display: flex;
  flex-flow: column wrap;
  justify-content: space-between;
  width: 100%;
  max-width: 867px;
  margin: 25px 10px;
  height: calc(100% - 50px);
  border: var(--border);
  border-radius: 5px;
  background: var(--msger-bg);
  box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}

.msger-header {
  /* display: flex; */
  font-size: medium;
  justify-content: space-between;
  padding: 10px;
  text-align: center;
  border-bottom: var(--border);
  background: #eee;
  color: #666;
}

.msger-chat {
  flex: 1;
  overflow-y: auto;
  padding: 10px;
}
.msger-chat::-webkit-scrollbar {
  width: 6px;
}
.msger-chat::-webkit-scrollbar-track {
  background: #ddd;
}
.msger-chat::-webkit-scrollbar-thumb {
  background: #bdbdbd;
}
.msg {
  display: flex;
  align-items: flex-end;
  margin-bottom: 10px;
}

.msg-img {
  width: 50px;
  height: 50px;
  margin-right: 10px;
  background: #ddd;
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  border-radius: 50%;
}
.msg-bubble {
  max-width: 450px;
  padding: 15px;
  border-radius: 15px;
  background: var(--left-msg-bg);
}
.msg-info {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 10px;
}
.msg-info-name {
  margin-right: 10px;
  font-weight: bold;
}
.msg-info-time {
  font-size: 0.85em;
}

.left-msg .msg-bubble {
  border-bottom-left-radius: 0;
}

.right-msg {
  flex-direction: row-reverse;
}
.right-msg .msg-bubble {
  background: var(--right-msg-bg);
  color: #fff;
  border-bottom-right-radius: 0;
}
.right-msg .msg-img {
  margin: 0 0 0 10px;
}

.msger-inputarea {
  display: flex;
  padding: 10px;
  border-top: var(--border);
  background: #eee;
}
.msger-inputarea * {
  padding: 10px;
  border: none;
  border-radius: 3px;
  font-size: 1em;
}
.msger-input {
  flex: 1;
  background: #ddd;
}
.msger-send-btn {
  margin-left: 10px;
  background: rgb(0, 196, 65);
  color: #fff;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.23s;
}
.msger-send-btn:hover {
  background: rgb(0, 180, 50);
}
.msger-chat {
  background-color: #fcfcfe;
  background-image: url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='260' height='260' viewBox='0 0 260
260'%3E%3Cg fill-rule='evenodd'%3E%3Cg fill='%23dddddd' fill-
opacity='0.4'%3E%3Cpath d='M24.37 16c.2.65.39 1.32.54 2H21.17l1.17 2.34.45.9-
.24.11V28a5 5 0 0 1-2.23 8.94l-.02.06a8 8 0 0 1-7.75 6h-20a8 8 0 0 1-7.74-6l-
.02-.06A5 5 0 0 1-17.45 28v-6.76l-.79-1.58-.44-.9.9-.44.63-.32H-20a23.01
23.01 0 0 1 44.37-2zm-36.82 2a1 1 0 0 0-.44.1l-3.1 1.56.89 1.79 1.31-.66a3 3
0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0
0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .86.02l2.88-1.27a3 3 0 0 1
2.43 0l2.88 1.27a1 1 0 0 0 .85-.02l3.1-1.55-.89-1.79-1.42.71a3 3 0 0 1-
2.56.06l-2.77-1.23a1 1 0 0 0-.4-.09h-.01a1 1 0 0 0-.4.09l-2.78 1.23a3 3 0 0
1-2.56-.06l-2.3-1.15a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.1L.9 19.22a3 3 0 0
1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.1l-2.21 1.11a3 3 0 0
1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01zm0-2h-4.9a21.01 21.01 0 0 1 39.61
0h-2.09l-.06-.13-.26.13h-32.31zm30.35 7.68l1.36-.68h1.3v2h-36v-1.15l.34-.17
1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69
0L2.26 23h2.59l1.36.68a3 3 0 0 0 2.56.06l1.67-.74h3.23l1.67.74a3 3 0 0 0
2.56-.06zM-13.82 27l16.37 4.91L18.93 27h-32.75zm-.63 2h.34l16.66 5 16.67-.0
.214a29 29 0 0 0-57.97 0h57.96z'/%3E%3C/g%3E%3C/g%3E%3C/svg%3E");
}
```

## #templates/index.html (file name) #HTML Code

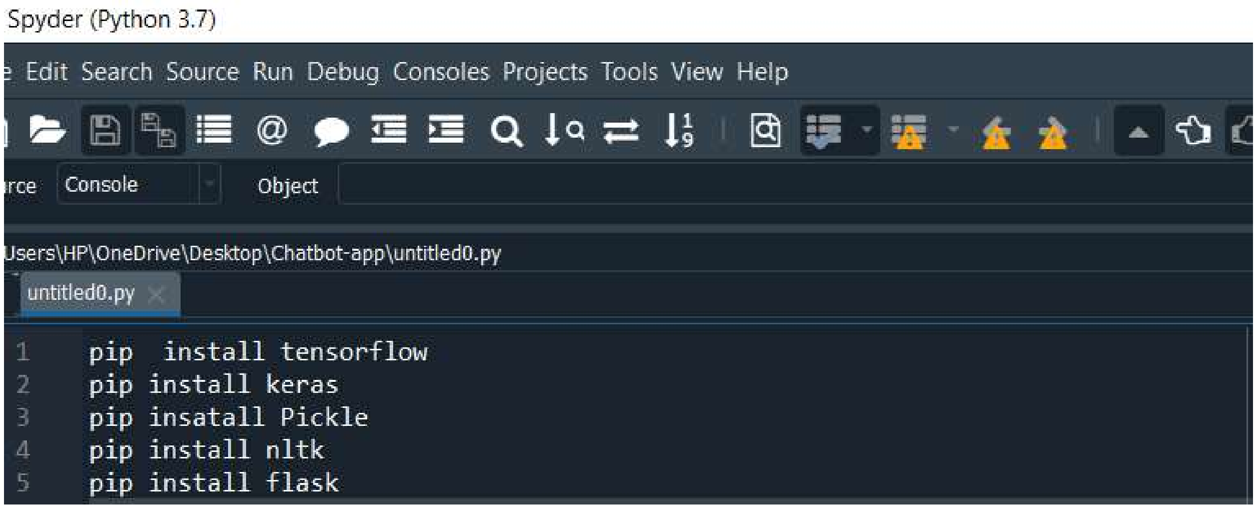
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Chatbot</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="{{ url_for('static',
filename='styles/style.css') }}">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></scri
pt>
</head>

<body>
  <!-- partial:index.partial.html -->
  <section class="msger">
    <header class="msger-header">
      <div class="msger-header-title">
        <i class="fas fa-bug"></i> Chatbot <i class="fas fa-bug"></i>
      </div>
    </header>

    <main class="msger-chat">
      <div class="msg left-msg">
        <div class="msg-img" style="background-image:
url(https://image.flaticon.com/icons/svg/327/327779.svg)"></div>

        <div class="msg-bubble">
          <div class="msg-info">
            <div class="msg-info-name">Chatbot</div>
            <div class="msg-info-time">12:45</div>
          </div>

          <div class="msg-text">
            Hi, welcome to ChatBot! Go ahead and send me a message. □
          </div>
        </div>
      </div>

    </main>

    <form class="msger-inputarea">
      <input type="text" class="msger-input" id="textInput"
placeholder="Enter your message...">
      <button type="submit" class="msger-send-btn">Send</button>
    </form>
  </section>
  <!-- partial -->
  <script
src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
  <script>
```

```
    const msgerForm = get(".msger-inputarea");
    const msgerInput = get(".msger-input");
    const msgerChat = get(".msger-chat");


    // Icons made by Freepik from www.flaticon.com
    const BOT_IMG = "https://image.flaticon.com/icons/svg/327/327779.svg";
    const PERSON_IMG = "https://image.flaticon.com/icons/svg/145/145867.svg";
    const BOT_NAME = "    ChatBot";
    const PERSON_NAME = "You";

    msgerForm.addEventListener("submit", event => {
      event.preventDefault();

      const msgText = msgerInput.value;
      if (!msgText) return;

      appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
      msgerInput.value = "";
      botResponse(msgText);
    });

    function appendMessage(name, img, side, text) {
      //   Simple solution for small apps
      const msgHTML = `
<div class="msg ${side}-msg">
  <div class="msg-img" style="background-image: url(${img})"></div>

  <div class="msg-bubble">
    <div class="msg-info">
      <div class="msg-info-name">${name}</div>
      <div class="msg-info-time">${formatDate(new Date())}</div>
    </div>

    <div class="msg-text">${text}</div>
  </div>
</div>
`;

      msgerChat.insertAdjacentHTML("beforeend", msgHTML);
      msgerChat.scrollTop += 500;
    }

    function botResponse(rawText) {

      // Bot Response
      $.get("/get", { msg: rawText }).done(function (data) {
        console.log(rawText);
        console.log(data);
        const msgText = data;
        appendMessage(BOT_NAME, BOT_IMG, "left", msgText);

      });

    }
    // Utils
    function get(selector, root = document) {
      return root.querySelector(selector);
    }

    function formatDate(date) {
      const h = "0" + date.getHours();
      const m = "0" + date.getMinutes();

      return `${h.slice(-2)}:${m.slice(-2)}`;
    }



  </script>

</body>

</html>
```

**#app.py(file name) #Python Code**

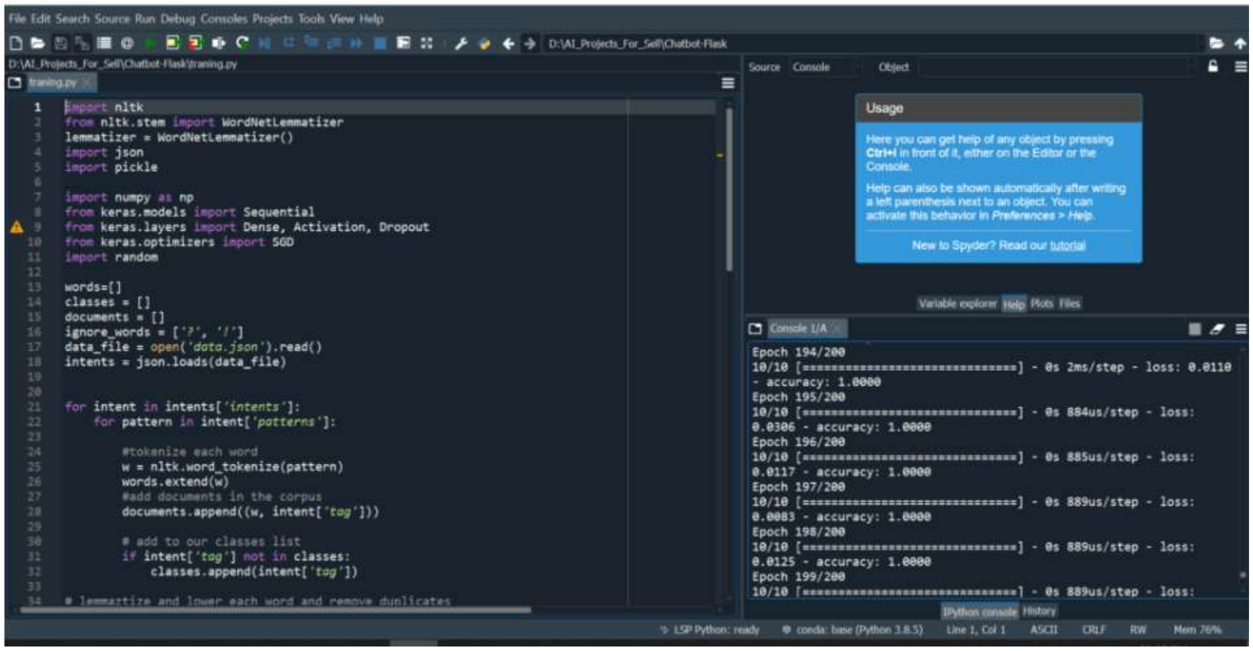
```
import nltk
nltk.download('popular')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('data.json').read())
words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.pkl','rb'))

def clean_up_sentence(sentence):

    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)

    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in 
sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in 
the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)

    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:

                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):

    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability":
str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res


from flask import Flask, render_template, request

app = Flask(__name__)
app.static_folder = 'static'

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)


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

#Note: First Run the training.py File to get the label.pkl (file) , model.h5
(file), text.pkl (file)

#After running Training.py File Then Run the app.py File to get the Output
(Chatbot interface)
```

## 5.2. SCREENSHOTS


Spyder (Python 3.7)
e Edit Search Source Run Debug Consoles Projects Tools View Help
rce Console Object
Users\HP\OneDrive\Desktop\Chatbot-app\untitled0.py
untitled0.py

```
pip  install tensorflow
pip install keras
pip insatall Pickle
pip install nltk
pip install flask
```

**#training.py(Outputs)**


File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\AI_Projects_For_Self\Chatbot-Flask
D:\AI_Projects_For_Self\Chatbot-Flask\traning.py
traning.py
Source Console Object
Usage
Variable explorer Help Plots Files
Console 1/A
IPython console History
LSP Python: ready conda: base (Python 3.8.5) Line 1, Col 1 ASCII CRLF RW Mem 76%

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('data.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmartize and lower each word and remove duplicates
```

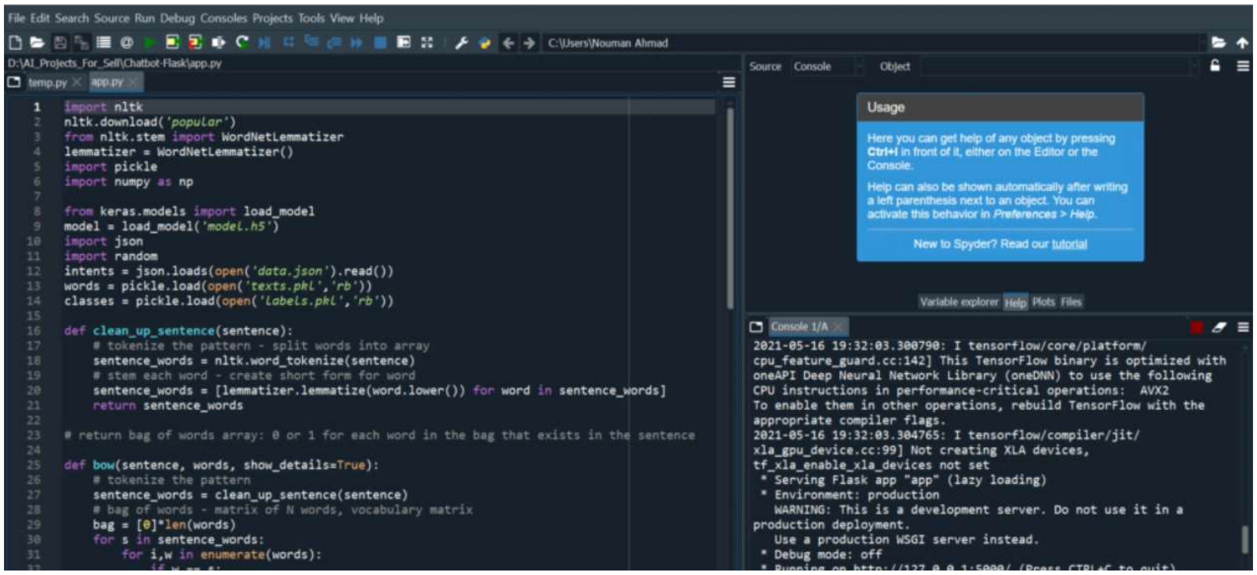
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

```
Epoch 194/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0110
- accuracy: 1.0000
Epoch 195/200
10/10 [==============================] - 0s 884us/step - loss:
0.0306 - accuracy: 1.0000
Epoch 196/200
10/10 [==============================] - 0s 885us/step - loss:
0.0117 - accuracy: 1.0000
Epoch 197/200
10/10 [==============================] - 0s 889us/step - loss:
0.0083 - accuracy: 1.0000
Epoch 198/200
10/10 [==============================] - 0s 889us/step - loss:
0.0125 - accuracy: 1.0000
Epoch 199/200
10/10 [==============================] - 0s 889us/step - loss:
```

## #app.py (Run Flask App)


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Nouman Ahmad
D:\AI_Projects_For_Self\Chatbot-Flask\app.py
temp.py
app.py
Source Console Object
Usage
Variable explorer Help Plots Files
Console 1/A

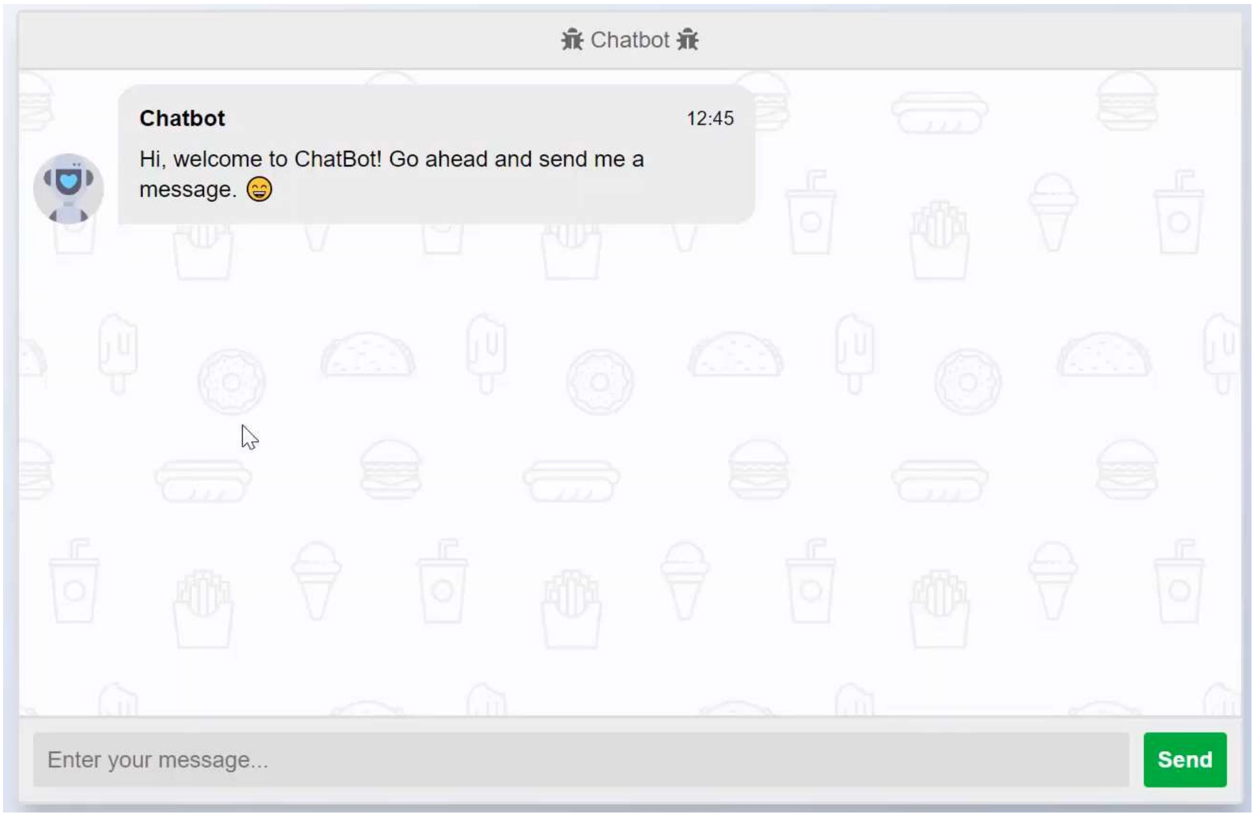
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

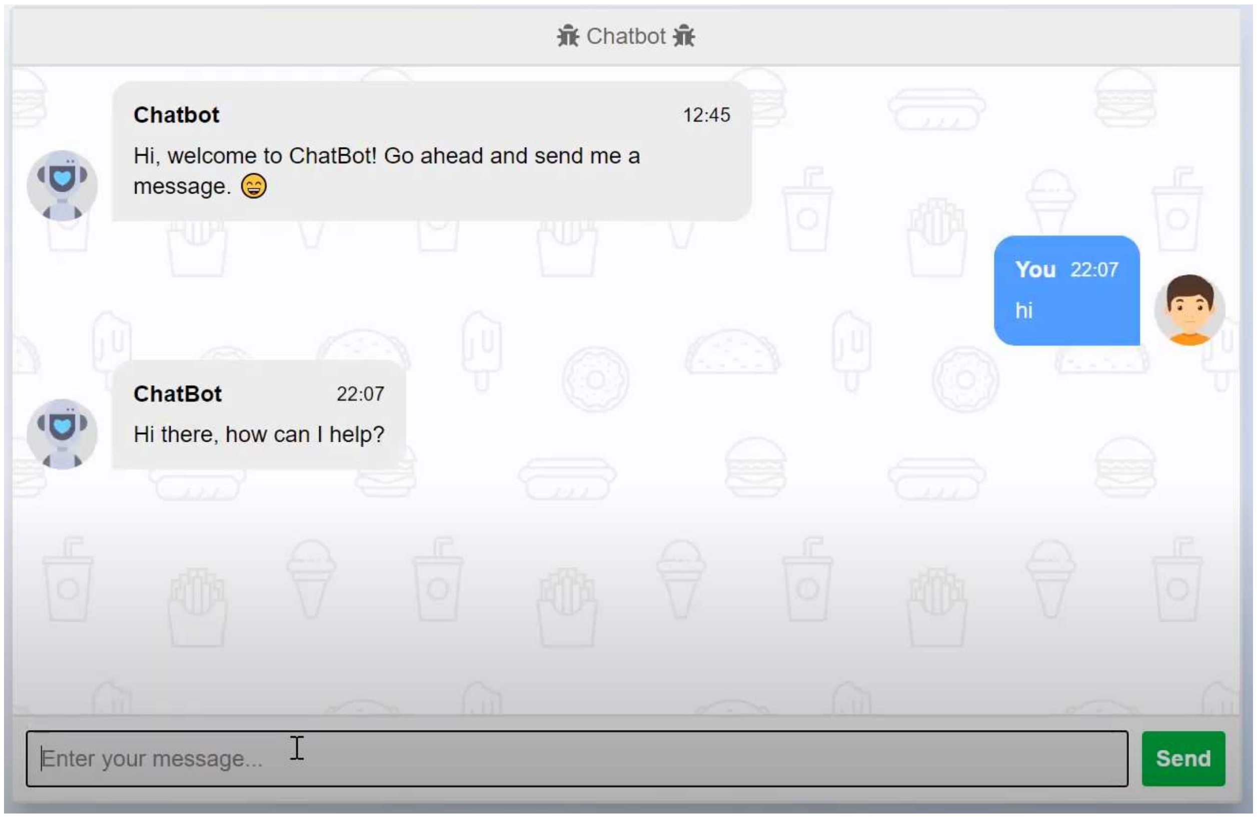
```python
import nltk
nltk.download('popular')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('data.json').read())
words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.pkl','rb'))

def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
```

```
2021-05-16 19:32:03.300790: I tensorflow/core/platform/
cpu_feature_guard.cc:142] This TensorFlow binary is optimized with
oneAPI Deep Neural Network Library (oneDNN) to use the following
CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the
appropriate compiler flags.
2021-05-16 19:32:03.304765: I tensorflow/compiler/jit/
xla_gpu_device.cc:99] Not creating XLA devices,
tf_xla_enable_xla_devices not set
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a
production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



**Chatbot** 12:45

Hi, welcome to ChatBot! Go ahead and send me a message. 😄

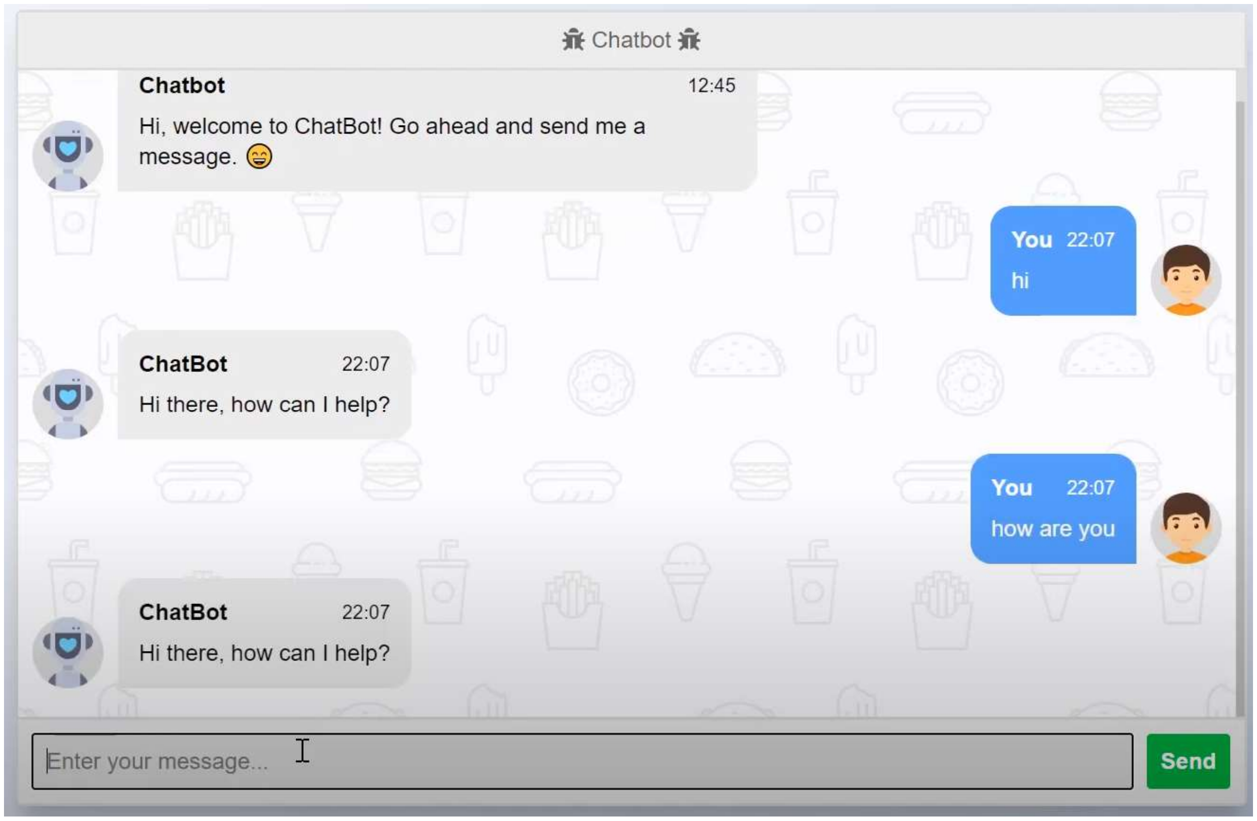


Chatbot

Chatbot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 22:07
hi

ChatBot 22:07
Hi there, how can I help?



Chatbot

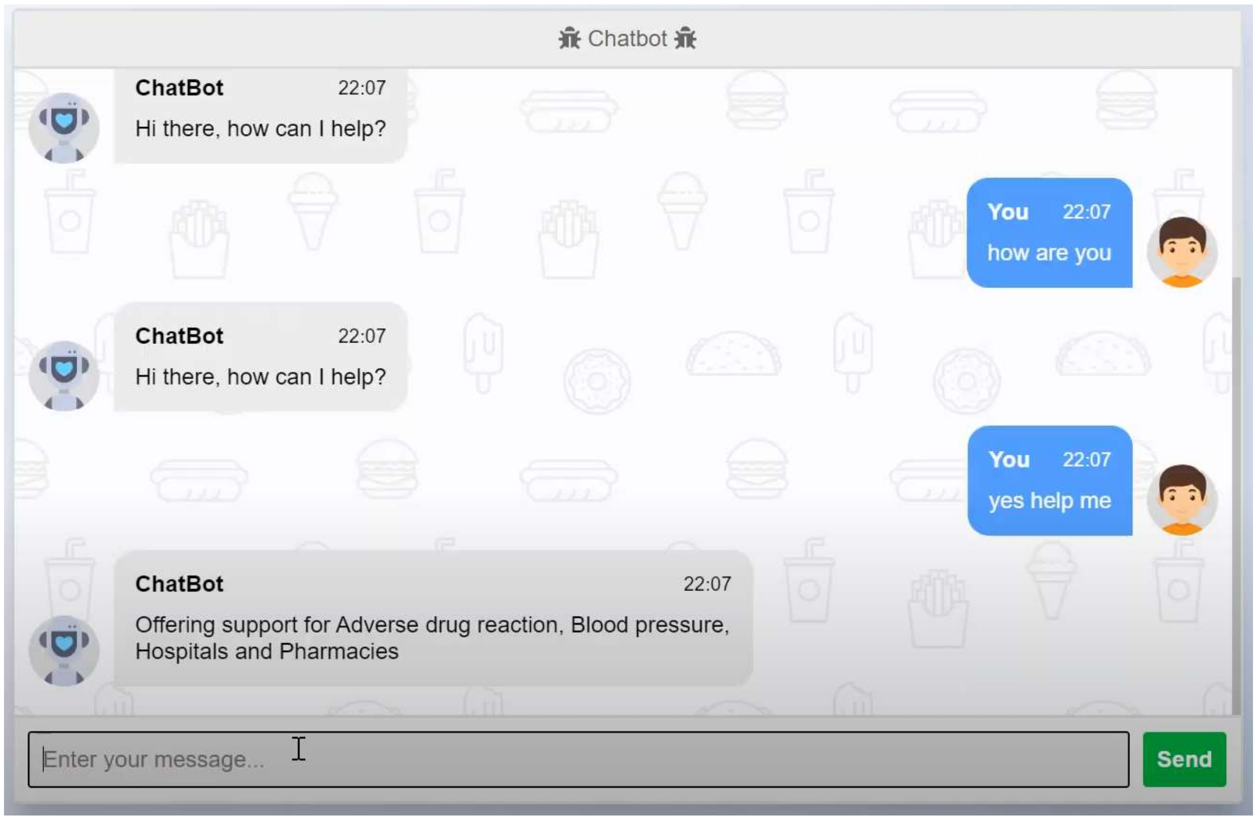
Chatbot 12:45
Hi, welcome to ChatBot! Go ahead and send me a message. 😄

You 22:07
hi

ChatBot 22:07
Hi there, how can I help?

You 22:07
how are you

ChatBot 22:07
Hi there, how can I help?



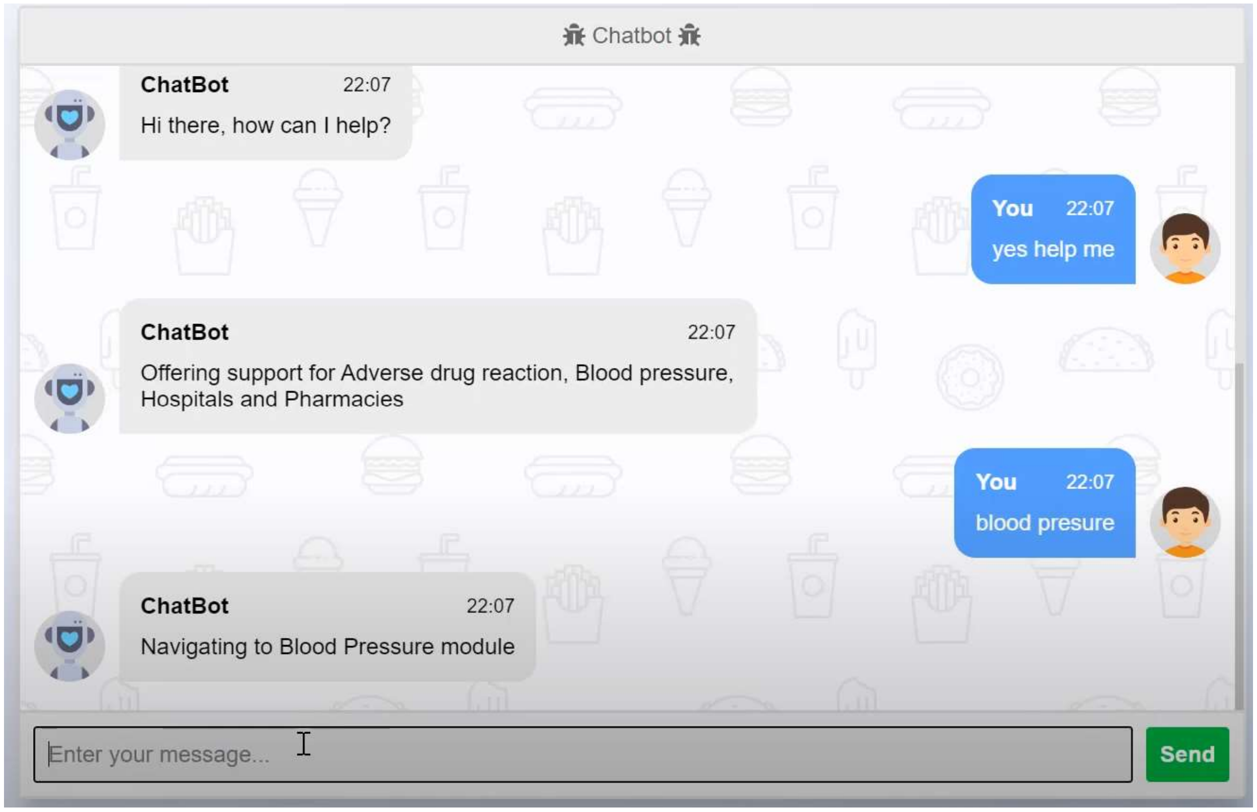
ChatBot 22:07
Hi there, how can I help?

You 22:07
how are you

ChatBot 22:07
Hi there, how can I help?

You 22:07
yes help me

ChatBot 22:07
Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies



ChatBot 22:07
Hi there, how can I help?

You 22:07
yes help me

ChatBot 22:07
Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies

You 22:07
blood presure

ChatBot 22:07
Navigating to Blood Pressure module

# 6. TESTING

**Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**TESTING IN STRATEGIES:**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two Strategies.

**Integrating Testing:**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing:**

Involves  in-house testing of the entire system before delivery to the user.  It aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing:**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

**Test Approach:**

Testing can be done in two ways:

Bottom up approach

Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled.  In case of erroneous input corresponding error messages are displayed.

# TEST CASES

| **Testcase** Id | Objective | Description | Expected Result |
|---|---|---|---|
| log001 | To test the URL | 1)Enter URL 2)Open the webpage 3)Check for GUI | Login validation page should open |
| log002 | To test the GUI | 1)open the webpage 2) check the GUI components | It should display webpage with label,email,password,submit…etc |
| log003 | To validate the Login page | 1)Open the login page 2)keep empty the input elements 3)click on login button | It should display error message as "Please enter email and password". |
| log004 | To validate the login page | 1)Open the login page 2)Enter valid credentials 3)click on login button | It should display message as "Login successfully ". |
| mv005 | To validate the mobile number | 1)Open the login page 2)Enter mobile with less than 10 digits 3)click on login button | It should display error message as "Please enter mobile number with 10 digits". |
| mv006 | To validate the mobile number | 1)Open the login page 2)Enter mobile with greater than 10 digits 3)click on login button | It should display error message as "Please enter mobile number with 10 digits". |
| mv007 | To validate the mobile number | 1)Open the login page 2)Enter mobile number starts with 6,7,8,9 digits 3)click on login button | It should sent OTP to the respective mobile number |
| mv008 | To validate the mobile number | 1)Open the login page 2)Enter mobile number doesnot starts with 6,7,8,9 digits 3)click on login button | It should display error message as "Please enter mobile number strats with 6,7,8,9 digits". |

|  |  |  |  |
|---|---|---|---|
| mv009 | To validate the mobile number | 1)Open the login page<br>2)Enter mobile number with special characters<br>3)click on login button | It should display error message as "Please enter mobile number with valid digits |
| mv010 | To validate the mobile number | 1)Open the login page<br>2)Enter mobile number with valid inputs<br>3)click on login button | It should sent OTP to the respective mobile number |

# 7. SCOPE

Quit close to be mainstream customer support to replace humans, chatbot have evolved by riding the artificial intelligence wave.

Every passing day they are becoming more intelligent and engaging. Inarguably the face of chatbot will undergo immense changes by 2020 but what could they be. Let us take closer look.

**Technological changes**

Artifical intelligence, programming languages, database and APIs are rapidly evolving and we can see their direct impact on chatbots.

**Chatbots are replacing apps**

The world has seen almost 6.5 million apps developed. It has been identified that 23% of the users uninstall the apps after a few weeks of use.

Enterprises are losing a lot of business this way. Chatbots come across as a potential way to engage with the audience then. A messaging app brings along a chatbot that is easy to download and user get engaged with the campaigns quickly.

**Emotional processing**

Advanced chatbots are just a fine line away from being capable of offering the human-level conversation. They are have already won over the natural language processing territory and next in scope is Natural Language Understanding (NLU) and Natural Language Generation(NLG).

**Voice Bots**

Chatbot's have received a lot of criticism from a section of developers who believe why to have a user a user interface. They prefer voice over taps. The development of smart speakers has been a step into this territory.

As per the predictive analysis (not by chatbots), soon you can see the chatbots making the place in Blockchain, Customer insight analysis, Banking and financial, Healthcare, Retail, Legal industry.

# 8. CONCLUSION

The  development costs of chatbots are getting cheaper. More and more industries wish to tap into the potential of the same and deliver enhanced customer experience. We expected certain industries easily adopting chatbots but a few have shown resistance. Inarguably the development world is working on top priority to remove the barriers to major mainstream adoption and we will soon be witnessing employees being augmented by efficient and quick chatbots.

# 9. BIBLIOGRAPHY

[1] **Akhil Mittal , Getting Started With Chatbots**

Learn and Create Your Own Chatbot With Deep Understanding Of Artificial Intelligence And Machine Learning.

[2]**Sebastian Raschka , Vahid Mirjalili , Python Machine Learning**

Machine Learning and Deep Learning with Python, scikit-learn and TensorFlow – Second Edition.

# 10.WEBLIOGRAPHY

Getting Started With Chatbots | Libraywala (librarywala.com)

Python Machine Learning - Second Edition | Libraywala (librarywala.com)

Google

YouTube