**CHATBOT USING PYTHON AND MACHING LEARNING**

**MINI PROJECT REPORT**

**Submitted by**

**S SAI DINEESHA (212219040126)**

**C H SAI SONICA (212219040130)**

**S SAI GAYATRI (212219040127)**

**In partial fulfilment of the award of the degree of**

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

****

**SAVEETHA ENGINEERING COLLEGE (AUTONOMOUS)**

**ANNA UNIVERSITY::CHENNAI-600 025**

**MARCH 2022**

**ANNA UNIVERSITY,CHENNAI**

**BONAFIDE CERTIFICATE**

Certified that this Mini Project report “CHATBOT” is the bonafide work of S.SAI DINEESHA (21221040126), S.SAI GAYATRI (212219040127), C.H.SAI SONICA (212219040130), who carried out the mini project work under my supervision.

**Signature Signature**

**Dr.R.KALADEVI,ME,PH.D Dr.G.NAGAPPAN,M.E,Ph.D**

Professor Professor

SUPERVISOR HEAD OF THE DPT Dept of Computer Science Dept of CSE,

and Engineering, Saveetha engineering clg

Saveetha Engineering College, Thandalam,Chennai,

Thandalam,Chennai,602105. 602105.

DATE OF THE VIVA VOICE EXAMINATION:……………

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

Chatbot is a vivid human-like contrivance which bounces chat. The way a machine can know human chats and how they response to the clients is a stimulating effort. Chatbot by means of neutral-network and deep learning models give tremendously good consequences in carrying a human verbal exchange. There are many present chats models; however, they have some problems. So, a bot can hardly replace a human, but it is a great help to accomplish specific objectives with a limited reach.  From the first chatbot to be created ELIZA to Amazon’s ALEXA today, chatbots have come a long way. This project is about building a chatbot using python.

**ACKNOWLEDGEMENT**

We express our deep sense of gratitude to our honorable and beloved Founder President Dr. N. M. Veeraiyan, our President Dr. Saveetha Rajesh, our Director Dr. S. Rajesh and other management members for providing the infrastructure needed.

We express our wholehearted gratitude to our principal, Dr. N. Duraipandian, for his wholehearted encouragement in completing this project.

We convey our thanks to Dr. G. Nagappan, Professor and Head of the Department of Computer Science and Engineering, Saveetha Engineering College, for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks and deep sense of gratitude to our Supervisor Dr. R. Kala Devi, Associate Professor, Department of Computer Science and Engineering, Saveetha Engineering College ,for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of the project work and for providing us necessary support and details at the right time and during the progressive reviews.

We owe our thanks to all the members of our college, faculty, staff and technicians for their kind and valuable cooperation during the course of the project. We are pleased to acknowledge our sincere thanks to our beloved parents, friends and well-wishers who encouraged us to complete this project successfully.

**Table of contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter Number** | **Title** | | **Page Number** |
|  | **Abstract** | |  |
|  | **INTRODUCTION** |
|  | 1.1 Overview of the project |
| **1** | 1.2 Scope and Objective | | **8** |
|  | **LITERATURE SURVEY** | |  |
|  | 2.1 Introduction | | 8 |
| **2** | 2.2 Literature Survey | | **10** |
|  | **SYSTEM DESIGN** | |  |
|  | 3.1 Introduction | | 13 |
| **3** | 3.2 Existing System | | **14** |
|  | 3.3 Proposed System | | 15 |
|  | 3.4 Algorithm | | 16 |
|  | 3.5 System Architecture Diagram | | 17 |
|  | 3.6 Data Flow Diagram | | 18 |
|  | 3.7 UML Use Case Diagram | | 20 |
|  | 3.8 Activity Diagram | | 21 |
|  | 3.9 System Requirements | | 24 |
|  | **IMPLEMENTATION AND ANALYSIS** | |  |
|  | 4.1 Python Library | | 24 |
| **4** | 4.2 Data | | **26** |
|  | 4.2.1 Getting the data | |  |  |
|  | 4.3 Feasibility Study | | 29 |  |
|  | 4.4 Module Description | | 31 |  |
|  | 4.5 Result and Analysis | | 36 |  |
|  | 4.6 System Design and Testing Plan | | 38 |  |
|  | 4.6.1 Types of testing | | 38 |  |

|  |  |  |
| --- | --- | --- |
| **5** | **Conclusion** |  |
|  | 5.1 Conclusion | 39 |
|  | 5.2 Future Enhancement | 39 |
|  | **Appendix A- Software Description** | **40** |
|  | **Appendix B- Sample Coding** | **42** |
|  | **Appendix C- Sample Output** | **46** |
|  | **References** | **47** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |
|  |
|  |
|  |
|  |

**` CHAPTER** **1**

**INTRODUCTION**

## **OVERVIEW OF THE PROJECT**

Chatbots are nothing but applications that are used by businesses or other entities to conduct an automatic conversation between a human and an AI. These conversations may be via text or speech. Chatbots are required to understand and mimic human conversation while interacting with humans from all over the world.

## **1.2 SCOPE AND OBJECTIVE**

## **SCOPE**

Chatbots are fully functioning, semi-autonomous systems that can **assist customer service experiences and response time**. The future scope of chatbots could include many benefits for enterprises.

## **OBJECTIVE**

The purpose of our study is to make an interactive chatbot that uses speech recognition and responds to the user .

Chatbots make the task easy for people and companies to answer customer queries in a short amount of time.

There are different types of chatbots and this one searches web and answers simple questions. This project can also be developed to answer for various languages.

**CHAPTER 2**

# **LITERATURE SURVEY**

## **2.1 INTRODUCTION**

Literature survey is the most important step in the software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool.

Once the programmers start building the tool the programmers need a lot of external support. This support can be obtained from senior programmers, from books or from websites. Before building the system, the above considerations are taken into account for developing the proposed system. A major part of the project development sector considers and fully surveys all the required needs for developing the project. For every project, the Literature survey is the most important sector in the software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, manpower, economy, and company strength.

Once these things are satisfied and fully surveyed, then the next step is to determine the software specifications in the respective system such as what type of operating system the project would require, and what all the necessary software is needed to proceed with the next step such as developing the tools, and the associated operations.

**2.2 LITERATURE SURVEY**

[1] Namita Mhatre, Karan Motani, Maitri Shah, Swati Mali, “Donna Interactive Chat-bot acting as a Personal Assistant” in International Journal of Computer Applications (0975 – 8887) Volume 140 – No.10, April 2016.

[2] M. Dahiya, “A Tool of Conversation: Chatbot” in International Journal of Computer Sciences and Engineering, Volume-5, Issue-5.

[3] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky, “Software Bots” in IEEE Software

[4] “How Do Chatbots Work? A Guide to the Chatbot Architecture”,

[5] Gk\_, “Soul of the Machine: How Chatbots Work.”, https://medium.com/@gk\_/how-chat-bots-workdfff656a35e2

[6] Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In Proceedings of ICLR.

[7] Yun-Nung Chen, Dilek Hakanni-Tur, Gokhan Tur, Asli ¨ Celikyilmaz, Jianfeng Gao, and Li Deng. 2016. Syntax or semantics? knowledge-guided joint semantic frame parsing. In Proceedings of the 6th IEEE Workshop on Spoken Language Technology. pages 348–355.

[8] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pages 484–495.

[9] Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, ¨ Yun-Nung Chen, Jianfeng Gao, Li Deng, and YeYi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In Proceedings of Interspeech. pages 715–719.

[10] Sungjin Lee. 2014. Extrinsic evaluation of dialog state tracking and predictive metrics for dialog policy optimization. In 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue. page 310.

[11] Oliver Lemon and Xingkun Liu. 2007. Dialogue policy learning for combinations of noise and user simulation: transfer results. In Proc. SIGdial.

[12] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation

[13] Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In Proceedings of Interspeech. pages 2506–2510.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. Nature 518:529–533.

[15] Alexander I Rudnicky, Eric H Thayer, Paul C Constantinides, Chris Tchou, R Shern, Kevin A Lenzo, Wei Xu, and Alice Oh. 1999. Creating natural dialogs in the carnegie mellon communicator system. In Eurospeech.

[16] Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. IEEE transactions on audio, speech, and language processing 17(4):733–747.

[17] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2017. A networkbased end-to-end trainable task-oriented dialogue system. pages 438–449.

[18] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, PeiHao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pages 1711–1721.

[19] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In Proceedings of the SIGDIAL 2013 Conference. pages 404–413

[20] Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue. pages 1–10.

**CHAPTER 3**

# **SYSTEM DESIGN**

## **3.1 INTRODUCTION**

Design is a multi- step that focuses on data structure software architecture, procedural details, algorithm and interface between modules. The design process also translates the requirements into presentation of software that can be accessed for quality before coding begins. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. Until the 1990s, systems design had a crucial and respected role in the data processing industry. In the 1990s, standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering. The architectural design of a system emphasizes the design of the system architecture that describes the structure, behaviours and more views of that system and analysis. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science in general. Computer software design changes continuously as new methods; better analysis and border understanding evolved. Software design is at a relatively early stage in its revolution. Therefore, software design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However techniques for software designs do exist, criteria for design qualities are available and design notation can be applied.

## **3.2 EXISTING SYSTEM**

* This chatbot decipher to human text and respond to it .

## **DISADVANTAGES**

The list of snags that a chatbot hits every time whenever user tries ties to interact with it are:

1: Synonyms, homonyms, slang

2: Misspellings

3: Abbreviations

4: Complex punctuation rules

5: Accents, dialects and speech differences with the age and other issues of humans. ( eg. lisps, drawls, etc)

## **3.3 PROPOSED SYSTEM**

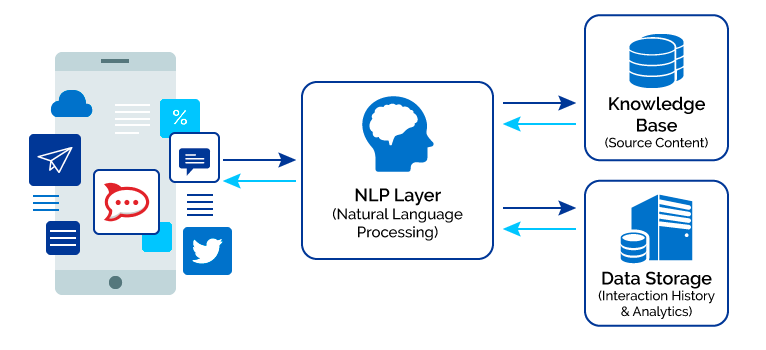
To build a simple chatbot that answers for simple questions.

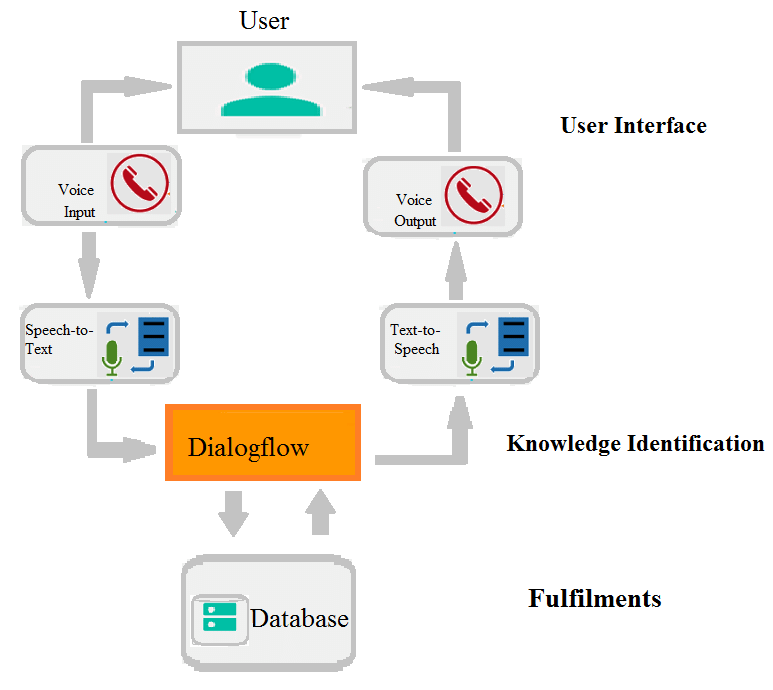
Tasks include preprocessing and training the bot. Other method is by searching the request in some web search and replying the first answer the search provides.

## **3.4 ALGORITHM**

NLP stands for Natural Language Processing. Using NLP technology, you can help a machine understand human speech and spoken words. NLP combines computational linguistics that is the rule-based modelling of the human spoken language with intelligent algorithms such as statistical, machine, and deep learning algorithms. These technologies together create the smart voice assistants and chatbots that you may be used in everyday life.

**3.5 SYSTEM ARCHITECTURE DIAGRAM**





**3.6 DATA FLOW DIAGRAM**

Data Flow Diagram (DFD) is a two-dimensional diagram that describes how data is processed and transmitted in a system. The graphical depiction recognizes each source of data and how it interacts with other data sources to reach a mutual output. In order to draft a data flow diagram one must

• Identify external inputs and outputs

• Determine how the inputs and outputs relate to each other

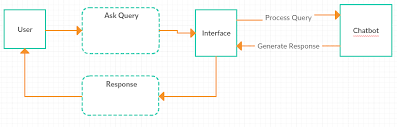
• Explain with graphics how these connections relate and what they result in.

**ROLE OF DFD**

• It is a documentation support which is understood by both programmers and nonprogrammers. As DFD postulates only what processes are accomplished not how they are performed.

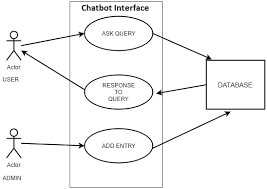
• A physical DFD postulates where the data flows and who processes the data.

• It permits analysts to isolate areas of interest in the organization and study them by examining the data that enter the process and viewing how they are altered when they leave



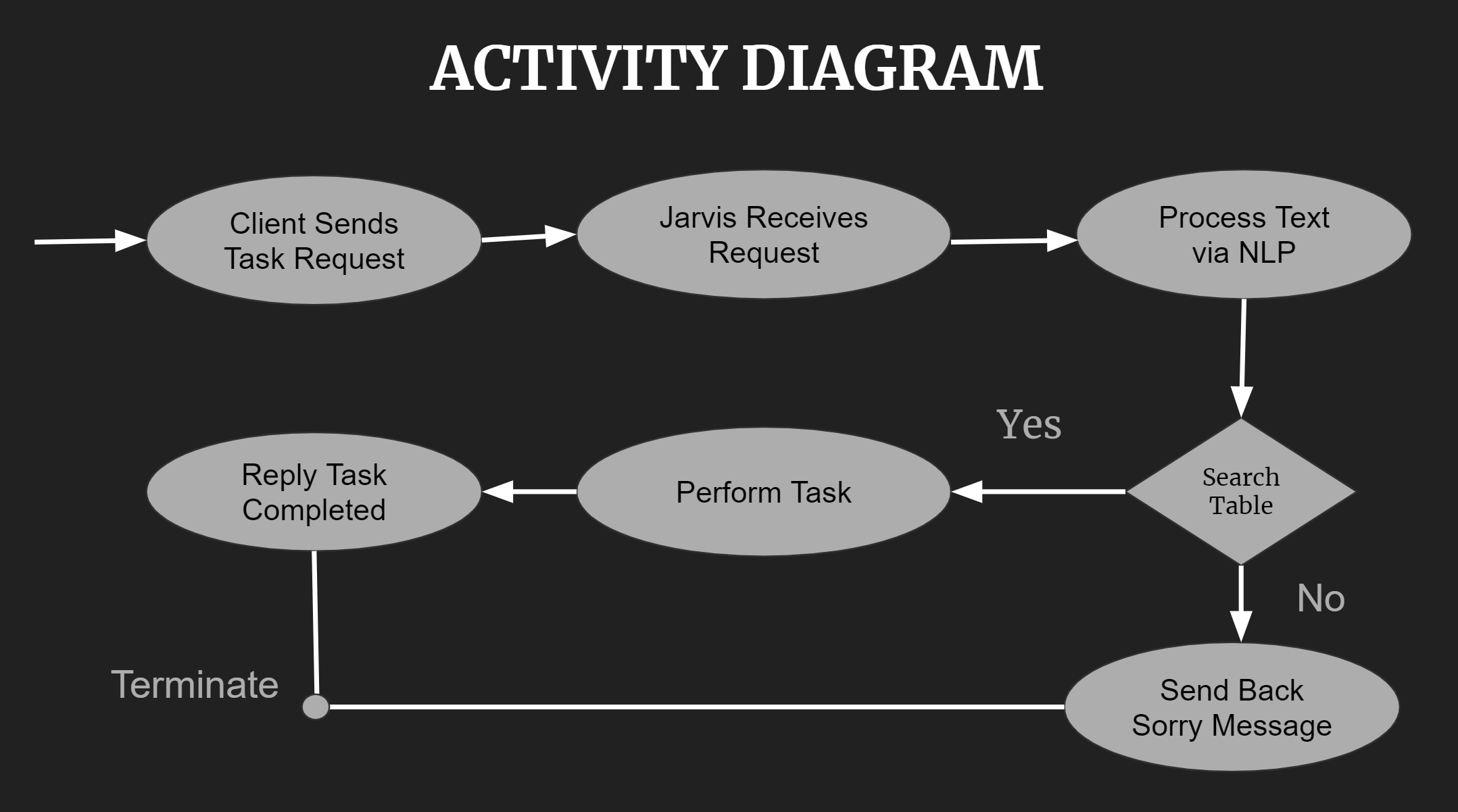
**3.7 UML DIAGRAMS USE CASE DIAGRAM**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.



**3.8 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities



**3.9 SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

• System - Pentium-IV

• Speed - 2.4 GHZ

• Hard disk - 40 GB

• Monitor - 15 VGA color

• RAM - 512 MB

**SOFTWARE REQUIREMENTS:**

• Operating System - Windows 10

• Coding Language - Python

• IDE - Jupyter Notebook/ Google collab

**CHAPTER 4**

**IMPLEMENTATION AND ANALYSIS**

**4.1 PYTHON LIBRARY**

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. The Python Standard Library is a collection of exact syntax, token, and semantics of Python. It comes bundled with core Python distribution. We mentioned this when we began with an introduction. For visualization of the dataset and prediction we use various python libraries.

Python library that is used in this project is NumPy

**Numpy:**

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow use NumPy internally for manipulation of Tensors.

**4.2 Data**

The crucial element in machine learning tasks for which a particular attention should be clearly taken is the data. Indeed the results will be highly influenced by the data.

**4.2.1 Getting the data**

The first step is to get the data from the user. Here data refers to the input that is spoken by the user. Based on this input the chatbot replies.

**IMPLEMENTATION**

**STEP 1: Create Virtual Environment**

pipenv is a python library to create virtual environment easily.

pip install pipenv

pipenv install

**STEP 2: Install Libraries**

We Will Use ChatterBot library to create Simple Python Chatbot.

Install chatterbot and chatterbot\_corpus with the help of pip command.

pipenv install chatterbot

pipenv install chatterbot\_corpus

**STEP 3: Create and Train the Chatbot**

from chatterbot import ChatBot

from chatterbot.trainers import ChatterBotCorpusTrainer

BOTNAME = "Pyter"

def start():

bot = ChatBot(BOTNAME,

logic\_adapters=[

{

'import\_path': 'chatterbot.logic.BestMatch',

'default\_response': 'I am sorry, but I do not understand.',

'maximum\_similarity\_threshold': 0.90,

},

],

preprocessors = [

"chatterbot.preprocessors.clean\_whitespace",

],

input\_adaptor="chatterbot.input.TerminalAdaptor",

output\_adaptor="chatterbot.output.TerminalAdaptor",

database\_uri='sqlite:///database.sqlite3')

trainer = ChatterBotCorpusTrainer(bot)

*# Train based on the english corpus*

trainer.train(

"chatterbot.corpus.english",

"chatterbot.corpus.english.greetings",

"chatterbot.corpus.english.conversations",

)

print(f"Hello I am {BOTNAME}")

while True:

try:

bot\_input = input("You: ")

bot\_respose = bot.get\_response(bot\_input)

print(f"{BOTNAME}: {bot\_respose}")

except(KeyboardInterrupt, EOFError, SystemExit):

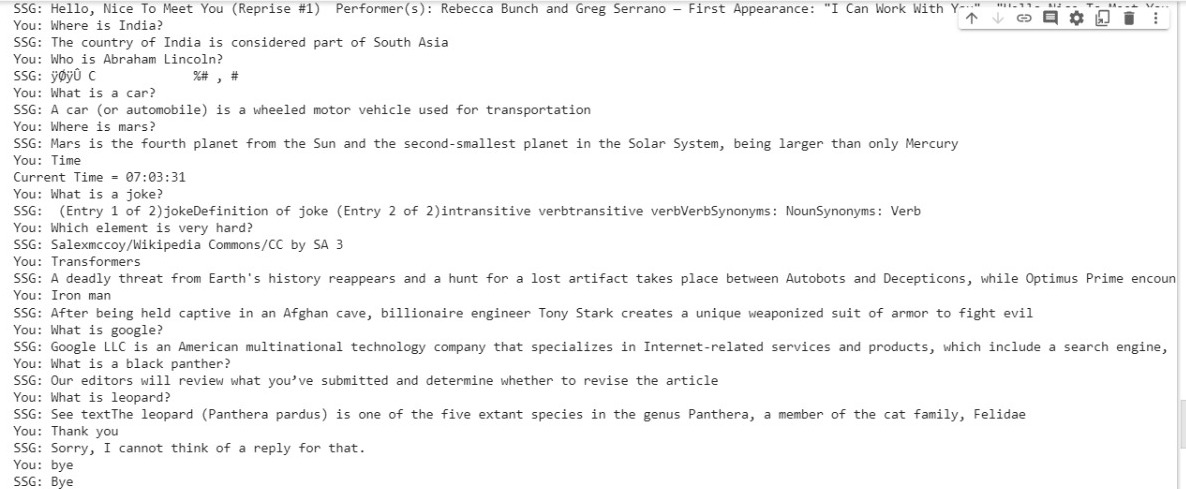
break

if \_\_name\_\_ == "\_\_main\_\_":

start()

**OUTPUT:**

****

****

**4.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

**4.3.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The system is an open source and doesn’t require any data collection.

**4.3.2 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is the Technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. Thus the developed system only needs preinstalled libraries and modules for the chatbot to run.

**4.3.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. This is a normal interactive chatbot. So it is socially acceptable .

**4.4 MODULE DESCRIPTION**

**MODULES**

**MODULE 1: SPEECH TO TEXT CONVERSION**

The first task that our chatbot must work for is the speech to text conversion. Basically, this involves converting the voice or audio signals into text data. In summary, the chatbot actually ‘listens’ to your speech and compiles a text file containing everything it could decipher from your speech.

**CODE:**

pip install pipenv

pipenv install

**MODULE 2 : RESPONSE**

Next, our AI needs to be able to respond to the audio signals that you gave to it. In simpler words, our chatbot has received the input. Now, it must process it and come up with suitable responses and be able to give output or response to the human speech interaction.

For this, the chatbot requires a text-to-speech module as well. Here, we will be using GTTS or Google Text to Speech library to save mp3 files on the file system which can be easily played back.

**CODE:**

pipenv install chatterbot

pipenv install chatterbot\_corpus

**MODULE 3 : UPGRADING CHATBOT**

Next, we can consider upgrading our chatbot to do simple commands like some o the virtual assistants help you to do. An example of such a task would be to equip the chatbot to be able to answer correctly whenever the user asks for the current time.

**CODE:**

from chatterbot import ChatBot

from chatterbot.trainers import ChatterBotCorpusTrainer

BOTNAME = "Pyter"

def start():

bot = ChatBot(BOTNAME,

logic\_adapters=[

{

'import\_path': 'chatterbot.logic.BestMatch',

'default\_response': 'I am sorry, but I do not understand.',

'maximum\_similarity\_threshold': 0.90,

},

],

preprocessors = [

"chatterbot.preprocessors.clean\_whitespace",

],

input\_adaptor="chatterbot.input.TerminalAdaptor",

output\_adaptor="chatterbot.output.TerminalAdaptor",

database\_uri='sqlite:///database.sqlite3')

trainer = ChatterBotCorpusTrainer(bot)

*# Train based on the english corpus*

trainer.train(

"chatterbot.corpus.english",

"chatterbot.corpus.english.greetings",

"chatterbot.corpus.english.conversations",

)

print(f"Hello I am {BOTNAME}")

while True:

try:

bot\_input = input("You: ")

bot\_respose = bot.get\_response(bot\_input)

print(f"{BOTNAME}: {bot\_respose}")

except(KeyboardInterrupt, EOFError, SystemExit):

break

if \_\_name\_\_ == "\_\_main\_\_":

start()

**MODULE 4: MAKING THE CHATBOT INTELLIGENT**

This is where the chatbot becomes intelligent and not just a scripted bot that will be ready to handle any test thrown at them. The main package that we will be using in our code here is the [Transformers](https://huggingface.co/transformers/) package provided by HuggingFace. This tool is popular amongst developers as it provides tools that are pre-trained and ready to work with a variety of tasks.

**CODE:**

**import requests**

**import string**

**from lxml import html**

**from googlesearch import search**

**from bs4 import BeautifulSoup**

**from chatterbot import ChatBot**

**from chatterbot.trainers import ChatterBotCorpusTrainer**

**from datetime import datetime**

**BOTNAME = "SSG"**

**def chatbot\_query(query, index=0):**

**fallback = 'Sorry, I cannot think of a reply for that.'**

**result = ''**

**try:**

**search\_result\_list = list(search(query, tld="co.in", num=10, stop=3, pause=1))**

**page = requests.get(search\_result\_list[index])**

**tree = html.fromstring(page.content)**

**soup = BeautifulSoup(page.content, features="lxml")**

**article\_text = ''**

**article = soup.findAll('p')**

**for element in article:**

**article\_text += '\n' + ''.join(element.findAll(text = True))**

**article\_text = article\_text.replace('\n', '')**

**first\_sentence = article\_text.split('.')**

**first\_sentence = first\_sentence[0].split('?')[0]**

**chars\_without\_whitespace = first\_sentence.translate(**

**{ ord(c): None for c in string.whitespace }**

**)**

**if len(chars\_without\_whitespace) > 0:**

**result = first\_sentence**

**else:**

**result = fallback**

**return result**

**except:**

**if len(result) == 0: result = fallback**

**return result**

**def start():**

**bot = ChatBot(BOTNAME,**

**logic\_adapters=[**

**{**

**'import\_path': 'chatterbot.logic.BestMatch',**

**'default\_response': 'I am sorry, but I do not understand.',**

**'maximum\_similarity\_threshold': 0.90,**

**},**

**],**

**preprocessors = [**

**"chatterbot.preprocessors.clean\_whitespace",**

**],**

**input\_adaptor="chatterbot.input.TerminalAdaptor",**

**output\_adaptor="chatterbot.output.TerminalAdaptor",**

**database\_url="https://www.cs.cmu.edu/~ark/QA-data/data/Question\_Answer\_Dataset\_v1.2.tar.gz")**

**trainer = ChatterBotCorpusTrainer(bot)**

**# Train based on the english corpus**

**trainer.train(**

**"chatterbot.corpus.english",**

**"chatterbot.corpus.english.greetings",**

**"chatterbot.corpus.english.conversations",**

**)**

**print(f"Hello I am {BOTNAME}")**

**now = datetime.now()**

**current\_time = now.strftime("%H:%M:%S")**

**while True:**

**try:**

**bot\_input = input("You: ")**

**bot\_respose = chatbot\_query(bot\_input)**

**if bot\_input=='Bye' or bot\_input =='bye':**

**print('SSG: Bye')**

**break**

**elif bot\_input=='Current time' or bot\_input=='Time' or bot\_input=='time':**

**print("Current Time =", current\_time)**

**else:**

**print(f"{BOTNAME}: {bot\_respose}")**

**except(KeyboardInterrupt, EOFError, SystemExit):**

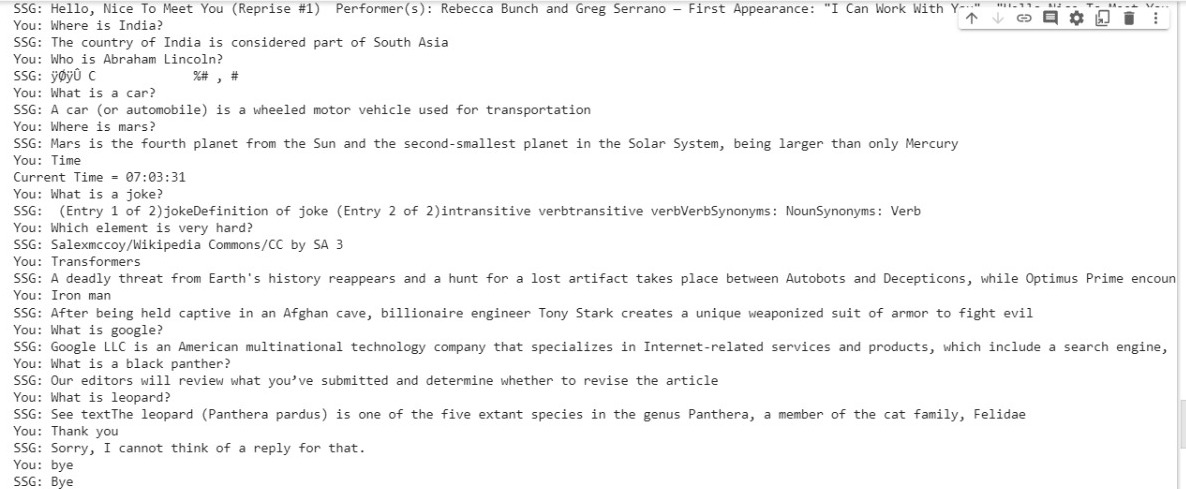
**break**

**if \_name\_ == "\_main\_":**

**start()**

**4.5 RESULT AND ANALYSIS:**

****



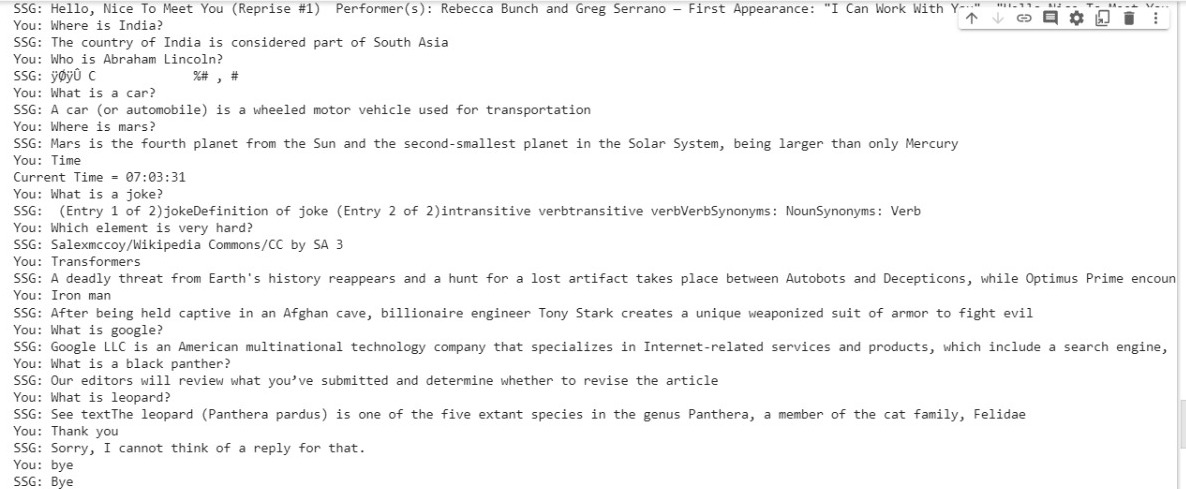
This is how the output appears. For this we have used Speech recognition, gTTS to convert speech to text and language models.

**4.6 SYSTEM DESIGN AND TESTING PLAN 4.6.1 Types of testing**

There are many types of testing like • Unit Testing • Integration Testing • Performance Testing

**4.6.1.1UnitTesting**  Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input. In Unit testing each module’s desired output is checked. As shown in the figure we can see that we get the desired results. And all the other modules are also giving the desired results.





**4.6.1.2IntegrationTesting**

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing. In this testing process all the modules in the system are integrated together and are tested for a given input and validates if the correct output is given.

**4.6.1.3PerformanceTesting**

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing. This system result depends on the user’s input, if their query is audible it will respond, else no output will be printed.

**CHAPTER 5 CONCLUSION**

**5.1 CONCLUSION**

To conclude, we have used Speech Recognition tools and NLP tech to cover the processes of text to speech and vice versa. Pre-trained Transformers language models were also used to give this chatbot intelligence instead of creating a scripted bot.

**5.2 FUTURE ENHANCEMENT**

This chatbot can be further enhanced to listen and reply as a human would.

This can be made modifications to create virtual assistant to integrate into business, project, or app support functions.

**APPENDIX A**

**SOFTWARE DESCRIPTION**

**A.1. PYTHON**

Python has huge set libraries which can be easily used for machine learning. Python is one of the languages which can be used to write codes in the Map-Reduce model while working in the Hadoop Ecosystem. Also, Spark, which is one of the new technologies for scalable big-data analysis, has a machine learning library (MLlib) in python. So, simplicity and wider applicability goes hand-in-hand to make it the so-called machine learning language. Python is an interpreter, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, it also has a comprehensive standard library. Python interpreters are available for many operating systems.

**APPENDIX B**

**SAMPLE CODING**

**CODE:**

**pip install pipenv**

**pipenv install**

**pipenv install chatterbot**

**pipenv install chatterbot\_corpus**

**from chatterbot import ChatBot**

**from chatterbot.trainers import ChatterBotCorpusTrainer**

**BOTNAME = "Pyter"**

**def start():**

**bot = ChatBot(BOTNAME,**

**logic\_adapters=[**

**{**

**'import\_path': 'chatterbot.logic.BestMatch',**

**'default\_response': 'I am sorry, but I do not understand.',**

**'maximum\_similarity\_threshold': 0.90,**

**},**

**],**

**preprocessors = [**

**"chatterbot.preprocessors.clean\_whitespace",**

**],**

**input\_adaptor="chatterbot.input.TerminalAdaptor",**

**output\_adaptor="chatterbot.output.TerminalAdaptor",**

**database\_uri='sqlite:///database.sqlite3')**

**trainer = ChatterBotCorpusTrainer(bot)**

**# Train based on the english corpus**

**trainer.train(**

**"chatterbot.corpus.english",**

**"chatterbot.corpus.english.greetings",**

**"chatterbot.corpus.english.conversations",**

**)**

**print(f"Hello I am {BOTNAME}")**

**while True:**

**try:**

**bot\_input = input("You: ")**

**bot\_respose = bot.get\_response(bot\_input)**

**print(f"{BOTNAME}: {bot\_respose}")**

**except(KeyboardInterrupt, EOFError, SystemExit):**

**break**

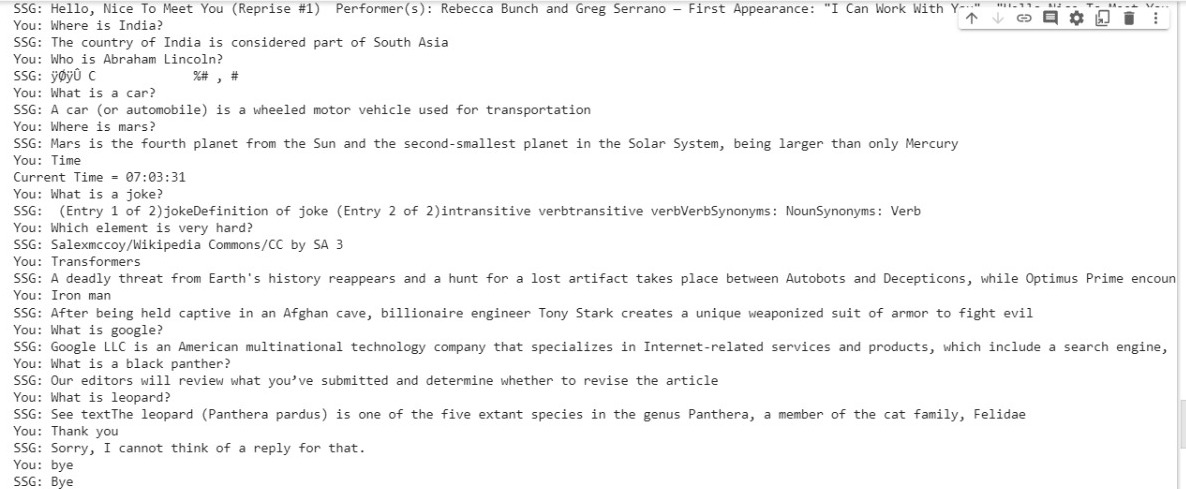
**if \_name\_ == "\_main\_":**

**start(**

**APPENDIX C**

**OUTPUT**

****



**REFERENCES:**

* **[1] Namita Mhatre, Karan Motani, Maitri Shah, Swati Mali, “Donna Interactive Chat-bot acting as a Personal Assistant” in International Journal of Computer Applications (0975 – 8887) Volume 140 – No.10, April 2016.**
* **[2] M. Dahiya, “A Tool of Conversation: Chatbot” in International Journal of Computer Sciences and Engineering, Volume-5, Issue-5.**
* **[3] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky, “Software Bots” in IEEE Software, www.computer.org/software.**
* **[4]“How Do Chatbots Work? A Guide to the Chatbot Architecture”**
* **[5] Gk\_, “Soul of the Machine: How Chatbots Work.”, https://medium.com/@gk\_/how-chat-bots-work-dfff656a35e2.**
* **[6] Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. IEEE transactions on audio, speech, and language processing 17(4):733–747.**
* **[7] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2017. A networkbased end-to-end trainable task-oriented dialogue system. pages 438–449.**
* **[8] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, PeiHao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pages 1711–1721.**
* **[9] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In Proceedings of the SIGDIAL 2013 Conference. pages 404–413**
* **[10] Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue. pages 1–10.**