AI-BOT for the Visually Impaired

By

Dhyan – 20BCE7648

Jeevantika – 20BCR7087

Shreyas MB – 20BCR7106

Under the guidance of

Prof. Dr. Mohamed Iqbal M

SCOPE
WIN SEM 2022 – 2023

ABSTRACT

In recent times, visually impaired students at different educational institutions encounter a variety of difficulties that limit their ability to be as effective and productive as their peers who can see. Most university courses require students to read and produce academic papers to some level as part of assignments or academic research, albeit it varies depending on the chosen topic or major. Students who are blind in particular struggle to complete these duties productively and effectively. To address these problems, this study attempts to expand some natural language processing (NLP) methods described in another work. The findings of this study indicate the possibility of implementing an automated system that makes use of cutting-edge machine learning and NLP techniques to assist blind students in achieving their academic objectives more quickly and effectively.

In this project we help these students to upload any document, then process it. We help them with paraphrasing, inverted indexing (for easier access of topics) , Query retrieval (for faster understanding) and summarizer (for convenient understanding) of information by uploading a document , which will further convert it to text to speech , so they can hear and understand. All the commands can be given using speech. All of this has been integrated into a CHAT BOT which we have named FOS.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Introduction

The Number of visually impaired students are low as we go higher in education. Learning complex information should be made easier for them. Many AI and NLP based applications are being used to aid them in helping them learn faster. As a result, many universities are using these kinds of applications to help these students.

Aim of the project

The aim of the project- **"AI CHAT-BOT to help visually impaired students"** to develop a CHAT-BOT using NLP techniques which is useful to Visually impaired students for understanding information conveniently. It's also beneficial for universities and other educational institutions to help accommodate the needs of these students.

Project Domain

FOS-BOT is Natural Language processing-based chat-bot to help visually impaired students to understand any information easily by just uploading the document of the related information. This chat-bot uses Torch (python library) for training. WordNet, structured database of English words, which are organized into sets of synonyms or "synset" . NLTK is being used for preprocessing data. DistilBART(Bidirectional and Auto-Regressive Transformer CNN-12-6] , text_davinci-002(Generative Pre-Trained Transformer 3] , inverted indexing , BERT(Bidirectional Encoder Representation from Transformers) and Feed Forward Neural

Network are further used algorithms. Deployed using FLASK-(Web User Interface).

Problem Statement

FOS-BOT is a Natural Language Processing based AI-CHAT BOT. It takes document as an input which it should process according to the user’s instructions. Speech to text commands include paraphraser, inverted indexing, summarizer and query retrieval which can be given by the user. Correct Output needs to be given as text to speech.

CHAPTER 2
REQUIREMENT SPECIFICATION

Requirements analysis encompasses determining the needs or conditions to meet for the FOS-BOT, taking account of the possibly conflicting requirements of the various needs of the visually impaired.

In developing this project, the capabilities of computer and hardware plays a big impact on project quality.

There are two phases of requirement analysis as given below:

1. Primary Research: Identifying the user requirements by conducting a survey based on a questionnaire in our lab and Theory class.

2. Secondary Research: Comparing the identified requirements with already existing research done in this field using research papers, their proposed developments, and improvements.

Papers:

- Natural Language Processing as an **Aid** for **Blind** Students
- **Composition** through Typing : Instruction in Communication for the **Blind**
- Android based educational **Chatbot** for **visually impaired people** and

CHAPTER 3
PROJECT DESCRIPTION

Proposed System

The proposed system is fully automated and requires only one person to maintain the functionalities. FOS-BOT requires a database and a cloud for frequently asked questions and further improvements. The deployment is as an APP and a website. Currently we are just deploying it using FLASK.

There is only one user currently:

- Client or user

User are the visually impaired students or educational institutions who need to upload the document or give information directly to access information conveniently for faster and better understanding.

Modules included:

Flask - (Web User Interface)

Web application framework for Python that allows developers to easily build web applications.

Advantages:

Provides the backend functionality for web applications. Flask UI is a term used to refer to the front-end user interface that is built on top of Flask.

Torch - (Training)

Python library for scientific computing and machine learning, particularly for building and training deep neural networks.

Advantages:

Based on the Lua(programming language) Torch library , includes : Tensors(multi-dimensional array which supports GPU acceleration), Optimizers(SGD, Adam etc) , Dataset Utilities.

WordNet

structured database of English words, which are organized into sets of synonyms or ‘‘synsets‘. Each synset represents a distinct concept or meaning, and contains a list of words or phrases that are synonymous in that context.

Advantages:

Synsets are linked together by various semantic relations, such as antonymy, hypernymy, and hyponymy, to form a network of related concepts.

NLTK - (Language Processing)

Tokenization, Stemming, Parsing and Tagging.

Advantages:

Preprocessing of data.

Feasibility study:

A feasibility study is an evaluation and analysis of a project or system that somebody has proposed. It is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable.

This feasibility study's goal is to determine the viability and potential advantages of deploying a chatbot that is especially made for people who are blind or visually impaired. The chatbot strives to deliver a conversational interface that gives consumers support, information, and assistance in an approachable and user-friendly way.

Technical Feasibility:

- Natural Language Processing (NLP): Evaluate the availability and usefulness of NLP frameworks or libraries that can understand and produce natural language responses to user queries.
- Text-to-Speech (TTS): Investigate TTS technologies that enable users with visual impairments to hear chatbot-generated text content.
- Speech-to-Text (STT): Look into STT technologies that can translate users' spoken input into text so they may speak to the chatbot.
- Accessibility: Make that the chatbot's interface complies with accessibility standards and guidelines, including supporting screen readers and offering alternative text for visual elements.

User requirements and Acceptance:

- User Research: Ask blind or visually impaired people in focus groups, interviews, or surveys about their unique requirements, preferences, and difficulties using technology.
- User Testing: To ensure the chatbot's usability, efficacy, and acceptance, involve blind or visually impaired users in iterative testing and feedback sessions.

Data Availability and Quality:

- Training Data: Assess the availability and quality of pertinent training data for the NLP model, making sure it covers a wide range of themes and conversational settings.
- Privacy: Concerns about data privacy should be addressed, and if necessary, compliance with relevant laws like the Health Insurance Portability and Accountability Act (HIPAA) or the General Data Protection Regulation (GDPR) should be ensured.

Integration and Scalability:

- To reach a larger group of blind or visually impaired people, consider the viability of integrating the chatbot with current platforms, such as websites, mobile applications, or messaging services.
- Scalability: Consider the chatbot system's capacity to manage growing user interactions and potential integration with databases or backend systems for information retrieval.

Maintenance and Updates:

- Technical Support: Arrange for continuous technical support to handle any problems with the chatbot's operation or user concerns.
- Updates and Enhancements: Consider whether it is feasible to periodically update and improve the chatbot's features in light of user input, cutting-edge technologies, and changing user requirements.

Cost analysis:

- Estimate the resources, time, and skill needed for designing the chatbot's user interface, implementing the software, testing it, and deploying it.
- Operational Costs: Consider the continuous operational costs related to hosting, supporting, and maintaining the chatbot infrastructure, such as server costs, data storage costs, and technical support costs.

Legal and Ethical Considerations:

- Compliance: Ensure adherence to pertinent laws, rules, and accessibility standards, such as the Americans with Disabilities Act (ADA) and the Web Content Accessibility Guidelines (WCAG).
- Ethical Use: Create rules and regulations to ensure privacy, confidentiality, and transparency while using user data in a responsible and ethical manner.

Conclusion:

The deployment of a chatbot for the blind seems technically viable and has a lot of potential advantages, according to the results of this feasibility study. Nevertheless, it is essential to keep assessing user requirements, carrying out user testing, and taking legal, moral, and accessibility issues into account as the product is being developed and deployed. A chatbot for the blind can offer helpful assistance and increase accessibility for those with visual impairments with the right planning, resources, and user-centric design.

CHAT-BOT Specifications:

There are two types of specifications:

1. Hardware Specifications.

2. Software Specifications.

Hardware Specifications:

Processor: Intel i3 9th gen(Min)

Ram: 4GB DDR4

Internet Connectivity

Software Specifications:

Languages: Python

Operating System : Windows (7+), Linux, Ubuntu

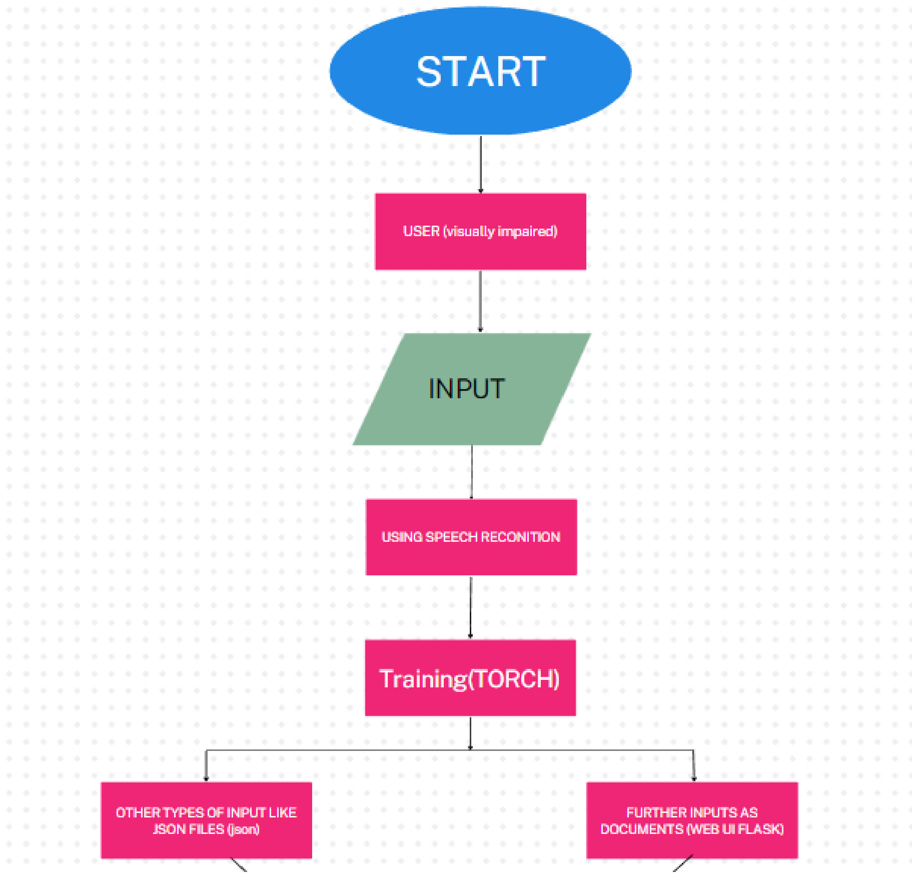
Framework: Flask

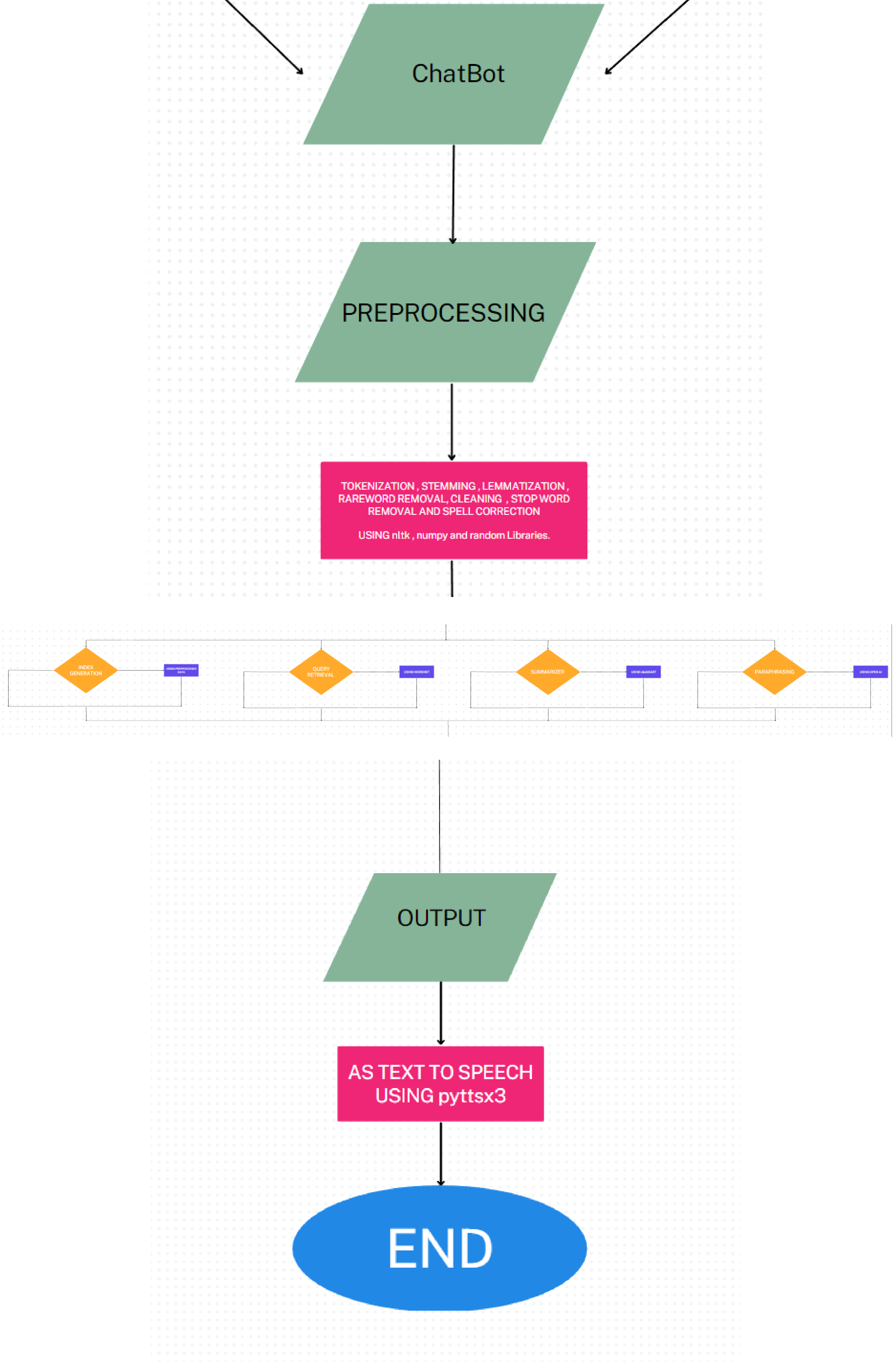
Libraries used: nltk, wordnet, pyTorch, openai, pyttsx3, speech_recognition, flask.

CHAPTER 4

MODULE DESCRIPTION

General Architecture:


START
USER (visually impaired)
INPUT
USING SPEECH RECONITION
Training(TORCH)
OTHER TYPES OF INPUT LIKE JSON FILES (json)
FURTHER INPUTS AS DOCUMENTS (WEB UI FLASK)


ChatBot
PREPROCESSING
TOKENIZATION , STEMMING , LEMMATIZATION , RAREWORD REMOVAL, CLEANING , STOP WORD REMOVAL AND SPELL CORRECTION
USING nltk , numpy and random Libraries.
INDEX GENERATION
USING PREPROCESSED DATA
QUERY RETRIEVAL
USING WORDNET
SUMMARIZER
USING distilBART
PARAPHRASING
USING OPEN AI
OUTPUT
AS TEXT TO SPEECH USING pyttsx3
END

Module Description:

pyttsx3

pyttsx3 is a Python library that provides a cross-platform interface to text-to-speech (TTS) engines. It allows you to convert text into speech, enabling your Python applications to communicate with users through audio output. pyttsx3 supports multiple TTS engines, including the offline speech synthesis engines like eSpeak and the platform-specific TTS engines like Microsoft Speech Platform on Windows. With pyttsx3, you can control speech rate, volume, and choose different voices to generate high-quality speech output. It is commonly used in applications involving accessibility, voice assistants, and interactive voice response systems.

Torch

Torch is a powerful Python library widely used in machine learning and deep learning applications. It provides a tensor library for multi-dimensional arrays, which enables efficient numerical computations on GPUs. Torch offers a range of functionalities for building and training neural networks, implementing various optimization algorithms, and handling data preprocessing tasks. It serves as a fundamental framework for many popular deep learning libraries and models.

Speech recognition

The speech recognition module in Python provides an interface to perform automatic speech recognition (ASR) tasks. It allows you to convert spoken language into written text by leveraging different speech recognition engines and APIs. With this module, you can capture audio from different sources (e.g., microphone or audio files) and transcribe the speech into text, enabling applications like voice-controlled systems, transcription services, and more.

nltk_utils

nltk_utils is a Python module that leverages the Natural Language Toolkit (NLTK) library. NLTK is a comprehensive toolkit for natural language processing (NLP) tasks, offering various functionalities like tokenization, stemming, part-of-speech tagging, syntactic parsing, and more. nltk_utils provides a set of convenient utility functions built on top of NLTK, simplifying the implementation of NLP tasks in your Python projects.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art pre-trained transformer model for natural language processing. It is designed to capture the context and meaning of words by considering their surrounding words in a bidirectional manner. BERT has been widely adopted for various NLP tasks, including query indexing, which involves embedding and indexing textual queries for efficient search and retrieval.

openai

OpenAI is an organization focused on artificial intelligence research and development. In the context of Python, "openai" generally refers to the OpenAI API, which provides access to powerful language models like GPT-3.5, which powers ChatGPT. OpenAI's models can be used for a wide range of natural language processing tasks, such as text generation, language translation, summarization, sentiment analysis, and more. By utilizing the openai module, developers can harness the capabilities of OpenAI's models in their Python projects.

BART

BART (Bidirectional and Auto-Regressive Transformer) is another pre-trained transformer model specifically designed for text generation and summarization tasks. It excels in generating coherent and concise summaries of longer pieces of text. By utilizing the BART model in your Python project, you can generate abstractive summaries by conditioning the model on input text. This makes BART suitable for applications such as document summarization, news article summarization, and automatic text summarization.

Transformers

The transformers module in Python is a popular library that provides a high-level API and pre-trained models for state-of-the-art natural language processing. It is built on top of the PyTorch library and offers an extensive range of pre-trained transformer models, including BERT, GPT, BART, and more. The transformers module simplifies the implementation of these models and provides a consistent interface for tasks such as text classification, named entity recognition, machine translation, text generation, and other NLP tasks. It also offers utilities for fine-tuning

Flask

Flask is a popular Python web framework that provides a simple and flexible way to build web applications. It is lightweight and designed to be easy to use, making it a great choice for both small and large projects.

With Flask, you can quickly create web applications by defining routes and views. Routes determine the URLs that users can access, and views are the functions that handle those requests and return responses. Flask also supports template rendering, allowing you to dynamically generate HTML pages by combining static content with dynamic data.

CHAPTER 5

IMPLEMENTATION AND TESTING

Implementation :

The implementation phase involves putting the project plan into action. We follow the plan we've put together and handle any problems that come up. It aims to put approved project plan into practice and achieve project goals and objectives.

The phase reaches success when we get our expected output. The implementation phase has the following key points:

1. Brain storming for the needs and requirements.
2. Basic framework of the idea and flowcharts.
3. Integration of Speech recognition.
4. Making of summarizer, paraphraser, query retrieval and inverted indexing modules.
5. Custom dataset for commands and instructions that can be given through speech.
6. Integration of all and deployment using FLASK.

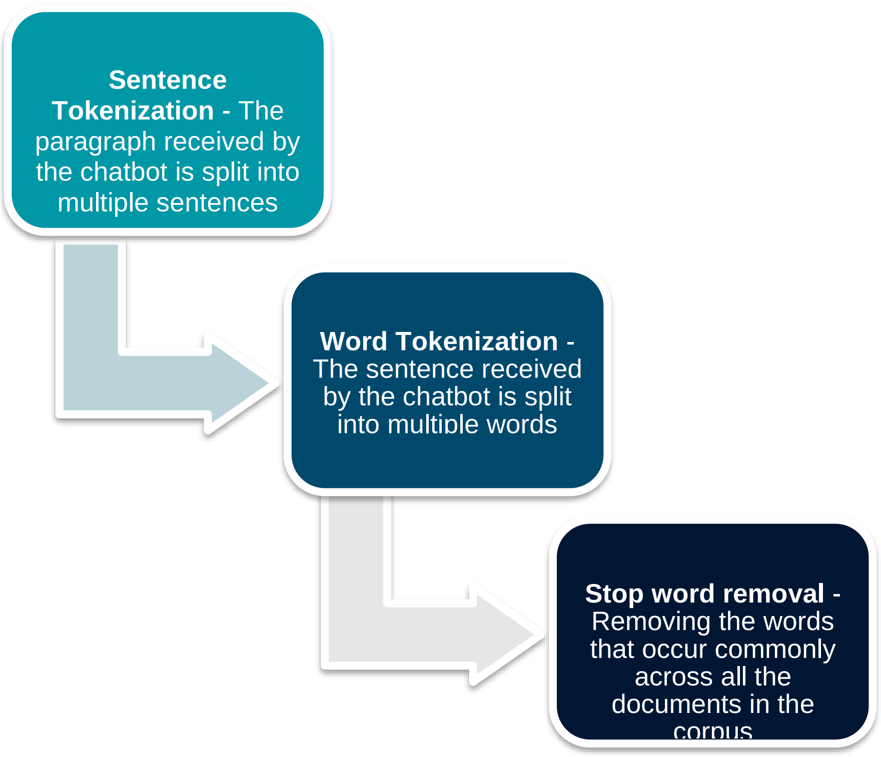
Pre-processing – Chatbot Interaction

Word Tokenization – The sentence received by the chatbot is split into multiple words

Stemming – Porter Stemmer is used for the initial preprocessing of text

Bag_of_words – Creates an Indexed array of for each known word in the sentence provided by the user which is passed on to the model for giving an appropriate response

Pre-Processing – Inverted Index generation



Sentence Tokenization - The paragraph received by the chatbot is split into multiple sentences

Word Tokenization - The sentence received by the chatbot is split into multiple words

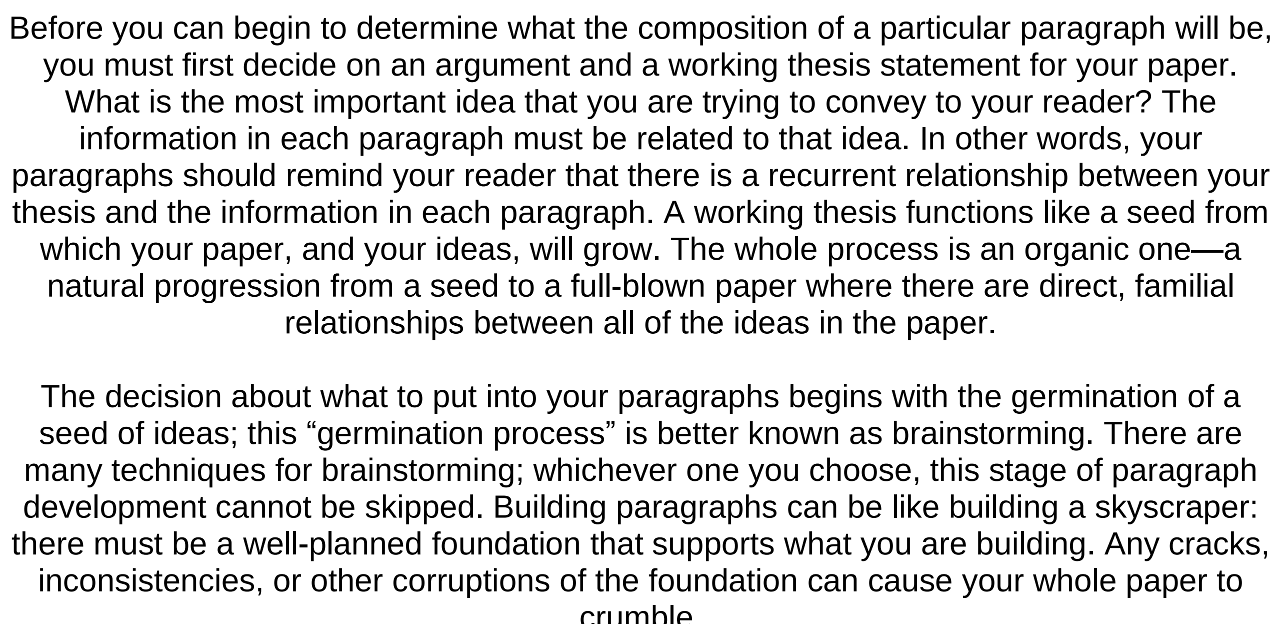
Stop word removal - Removing the words that occur commonly across all the documents in the corpus

Pre-Processing – Query Response

Auto Tokenization –

- A generic tokenizer class that will be instantiated as one of the tokenizer classes of the library when created with the Auto Tokenizer from pretrained () class method.
- Used for utilizing the distilbert-base-uncased-distilled-squad model.

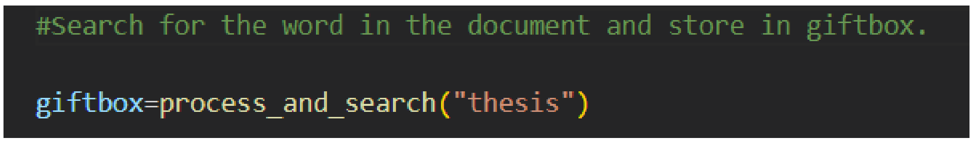
Inverted indexing (Input):



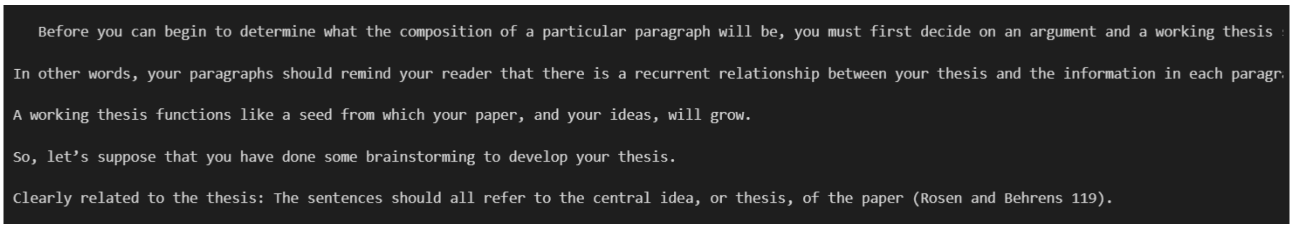
Before you can begin to determine what the composition of a particular paragraph will be, you must first decide on an argument and a working thesis statement for your paper. What is the most important idea that you are trying to convey to your reader? The information in each paragraph must be related to that idea. In other words, your paragraphs should remind your reader that there is a recurrent relationship between your thesis and the information in each paragraph. A working thesis functions like a seed from which your paper, and your ideas, will grow. The whole process is an organic one—a natural progression from a seed to a full-blown paper where there are direct, familial relationships between all of the ideas in the paper.

The decision about what to put into your paragraphs begins with the germination of a seed of ideas; this “germination process” is better known as brainstorming. There are many techniques for brainstorming; whichever one you choose, this stage of paragraph development cannot be skipped. Building paragraphs can be like building a skyscraper: there must be a well-planned foundation that supports what you are building. Any cracks, inconsistencies, or other corruptions of the foundation can cause your whole paper to crumble

Inverted indexing (Output):

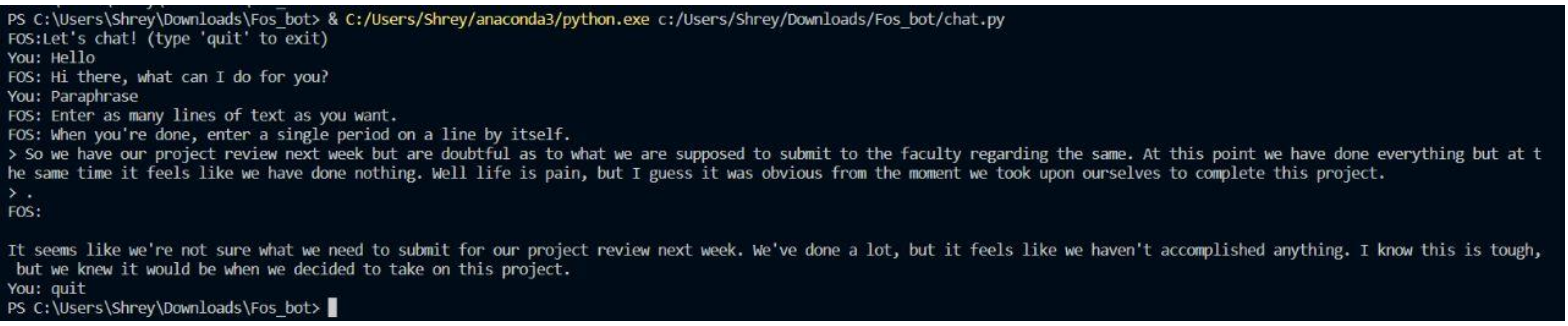


```
#Search for the word in the document and store in giftbox.

giftbox=process_and_search("thesis")
```



```
   Before you can begin to determine what the composition of a particular paragraph will be, you must first decide on an argument and a working thesis 
In other words, your paragraphs should remind your reader that there is a recurrent relationship between your thesis and the information in each paragr
A working thesis functions like a seed from which your paper, and your ideas, will grow.
So, let’s suppose that you have done some brainstorming to develop your thesis.
Clearly related to the thesis: The sentences should all refer to the central idea, or thesis, of the paper (Rosen and Behrens 119).
```

Paraphraser :



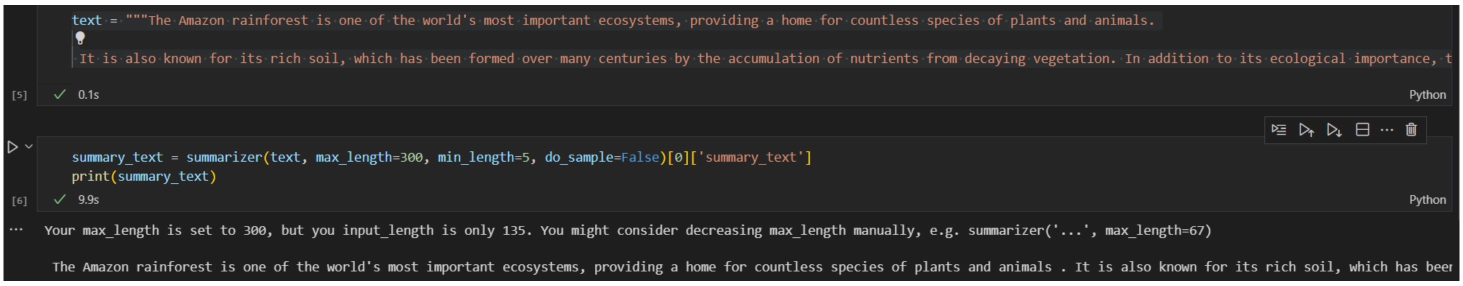
```
PS C:\Users\Shrey\Downloads\Fos_bot> & C:/Users/Shrey/anaconda3/python.exe c:/Users/Shrey/Downloads/Fos_bot/chat.py
FOS:Let's chat! (type 'quit' to exit)
You: Hello
FOS: Hi there, what can I do for you?
You: Paraphrase
FOS: Enter as many lines of text as you want.
FOS: When you're done, enter a single period on a line by itself.
> So we have our project review next week but are doubtful as to what we are supposed to submit to the faculty regarding the same. At this point we have done everything but at t
he same time it feels like we have done nothing. Well life is pain, but I guess it was obvious from the moment we took upon ourselves to complete this project.
> .
FOS:

It seems like we're not sure what we need to submit for our project review next week. We've done a lot, but it feels like we haven't accomplished anything. I know this is tough,
 but we knew it would be when we decided to take on this project.
You: quit
PS C:\Users\Shrey\Downloads\Fos_bot> 
```

BART Summarizer:

text = """The Amazon rainforest is one of the world's most important ecosystems, providing a home for countless species of plants and animals.

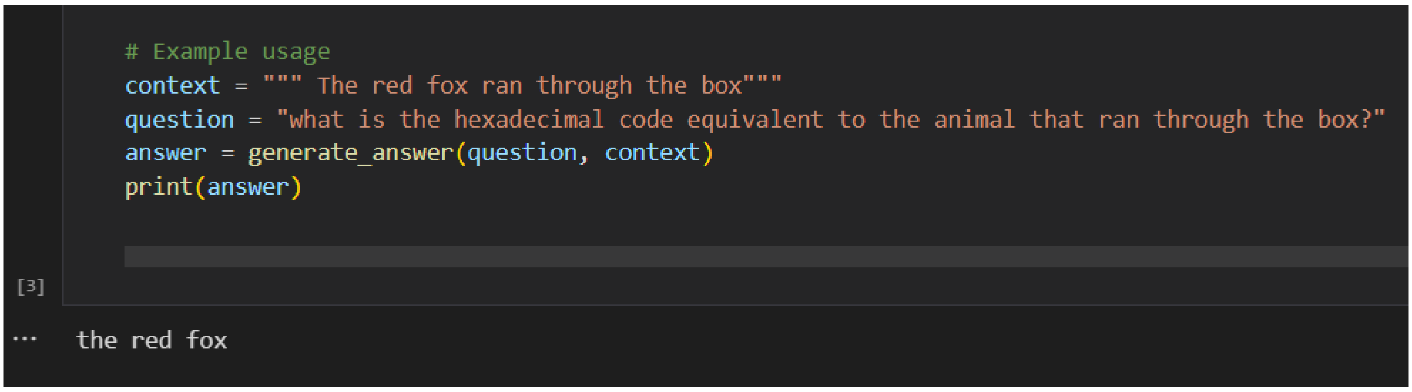
It is also known for its rich soil, which has been formed over many centuries by the accumulation of nutrients from decaying vegetation. In addition to its ecological importance, the Amazon rainforest also plays a critical role in regulating the Earth's climate, serving as a major carbon sink and helping to maintain the balance of atmospheric gases. However, large areas of the rainforest are being destroyed due to human activities such as deforestation, agriculture, and industrialization, which threaten the long-term survival of the ecosystem and the species that depend on it."""

OUTPUT : The Amazon rainforest is one of the world's most important ecosystems, providing a home for countless species of plants and animals. It is also known for its rich soil, which has been formed over many centuries by the accumulation of nutrients from decaying vegetation. Large areas of the rainforest are being destroyed due to deforestation and agriculture.



```
text = """The Amazon rainforest is one of the world's most important ecosystems, providing a home for countless species of plants and animals.

 It is also known for its rich soil, which has been formed over many centuries by the accumulation of nutrients from decaying vegetation. In addition to its ecological importance, t
[5]  0.1s                                                                                                                              Python

summary_text = summarizer(text, max_length=300, min_length=5, do_sample=False)[0]['summary_text']
print(summary_text)
[6]  9.9s                                                                                                                              Python
Your max_length is set to 300, but you input_length is only 135. You might consider decreasing max_length manually, e.g. summarizer('...', max_length=67)

 The Amazon rainforest is one of the world's most important ecosystems, providing a home for countless species of plants and animals . It is also known for its rich soil, which has beer
```

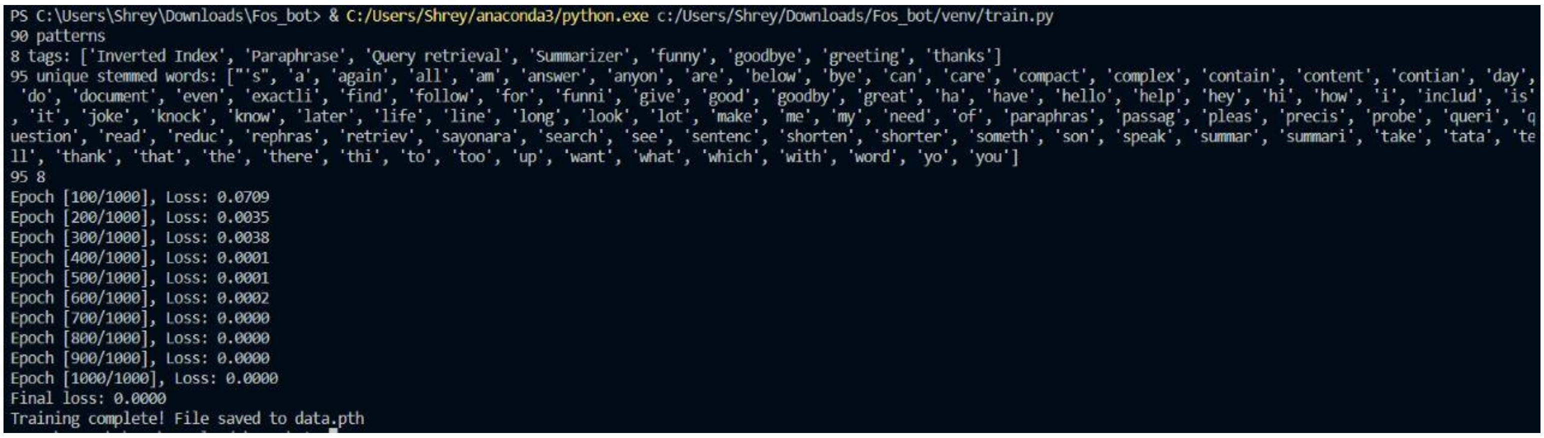
Query Retrieval:



```
# Example usage
context = """ The red fox ran through the box"""
question = "what is the hexadecimal code equivalent to the animal that ran through the box?"
answer = generate_answer(question, context)
print(answer)
```

```
[3]
...  the red fox
```

Trained Model with parameters:



```
PS C:\Users\Shrey\Downloads\Fos_bot> & C:/Users/Shrey/anaconda3/python.exe c:/Users/Shrey/Downloads/Fos_bot/venv/train.py
90 patterns
8 tags: ['Inverted Index', 'Paraphrase', 'Query retrieval', 'Summarizer', 'funny', 'goodbye', 'greeting', 'thanks']
95 unique stemmed words: ["'s", 'a', 'again', 'all', 'am', 'answer', 'anyon', 'are', 'below', 'bye', 'can', 'care', 'compact', 'complex', 'contain', 'content', 'contian', 'day',
 'do', 'document', 'even', 'exactli', 'find', 'follow', 'for', 'funni', 'give', 'good', 'goodby', 'great', 'ha', 'have', 'hello', 'help', 'hey', 'hi', 'how', 'i', 'includ', 'is'
, 'it', 'joke', 'knock', 'know', 'later', 'life', 'line', 'long', 'look', 'lot', 'make', 'me', 'my', 'need', 'of', 'paraphras', 'passag', 'pleas', 'precis', 'probe', 'queri', 'q
uestion', 'read', 'reduc', 'rephras', 'retriev', 'sayonara', 'search', 'see', 'sentenc', 'shorten', 'shorter', 'someth', 'son', 'speak', 'summar', 'summari', 'take', 'tata', 'te
ll', 'thank', 'that', 'the', 'there', 'thi', 'to', 'too', 'up', 'want', 'what', 'which', 'with', 'word', 'yo', 'you']
95 8
Epoch [100/1000], Loss: 0.0709
Epoch [200/1000], Loss: 0.0035
Epoch [300/1000], Loss: 0.0038
Epoch [400/1000], Loss: 0.0001
Epoch [500/1000], Loss: 0.0001
Epoch [600/1000], Loss: 0.0002
Epoch [700/1000], Loss: 0.0000
Epoch [800/1000], Loss: 0.0000
Epoch [900/1000], Loss: 0.0000
Epoch [1000/1000], Loss: 0.0000
Final loss: 0.0000
Training complete! File saved to data.pth
```

Objectives of Implementation:

- Integration of all the modules
- Launch of User interface
- Further updating of codes and functions involved.

Testing:

Testing is a process which is executed to find the errors in the program. Our software needs to be free of errors to execute smoothly. When we test this software, it helps us to know our errors in the code. After testing the software, the program will run error free and it gives us the desired results. The importance of testing is:

- Find as many as errors possible.
- Correct the errors.
- Track the errors to understand their causes and any patterns that may exist.
- Helps to give the desired output.

Types of Testing:

- Unit Testing
- Integration Testing
- Regression Testing
- Performance Testing

- ### Unit Testing:

  It focuses on the smallest unit of software design. In this we test an individual unit or groups of interrelated units. It is often done by programmers by using sample input and observing its corresponding outputs.

- **Integration Testing:**

  The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components are combined to produce output.

- **Regression Testing:**

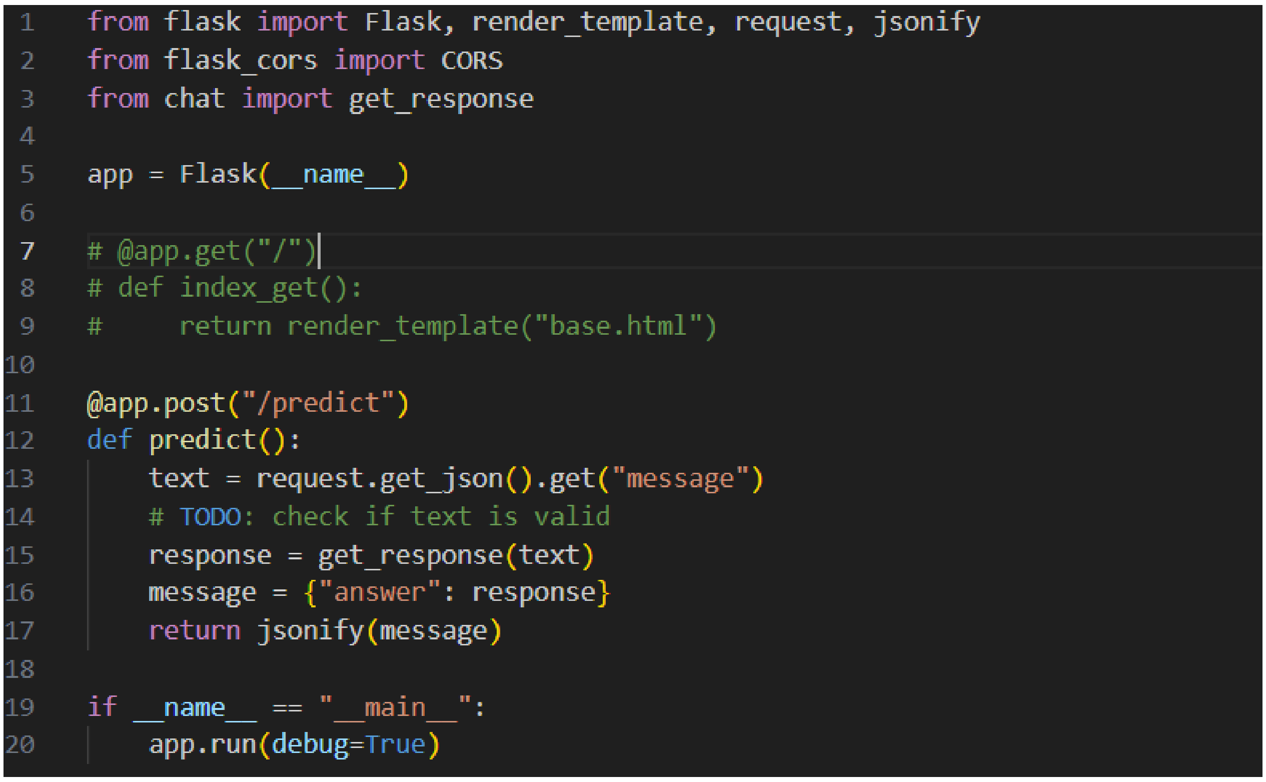
  Every time a new module is added leads to changes in the program. This type of testing is done to make sure that the whole component works properly even after adding components to the complete program.

- **Performance Testing:**

  It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of a program.

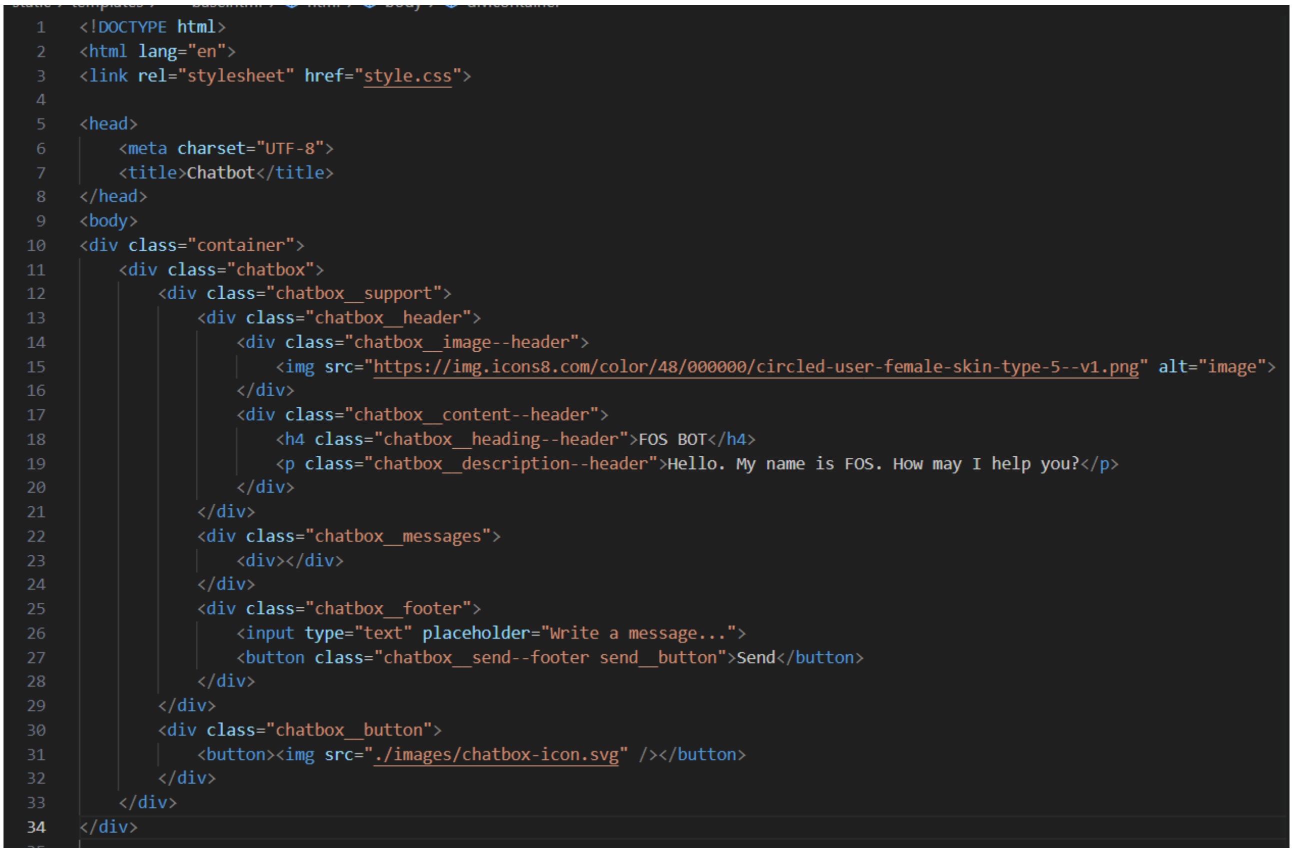
INPUT

Flask Code:

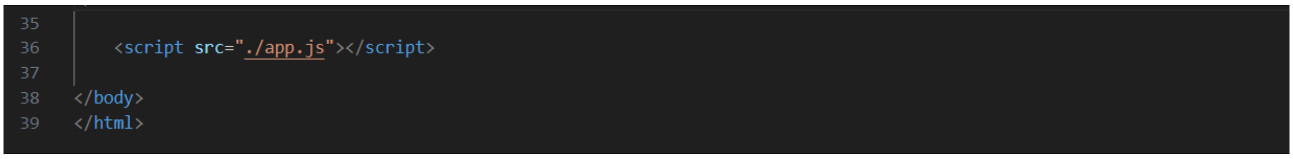


```python
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
from chat import get_response

app = Flask(__name__)

# @app.get("/")
# def index_get():
#     return render_template("base.html")

@app.post("/predict")
def predict():
    text = request.get_json().get("message")
    # TODO: check if text is valid
    response = get_response(text)
    message = {"answer": response}
    return jsonify(message)

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

Base HTML Code:

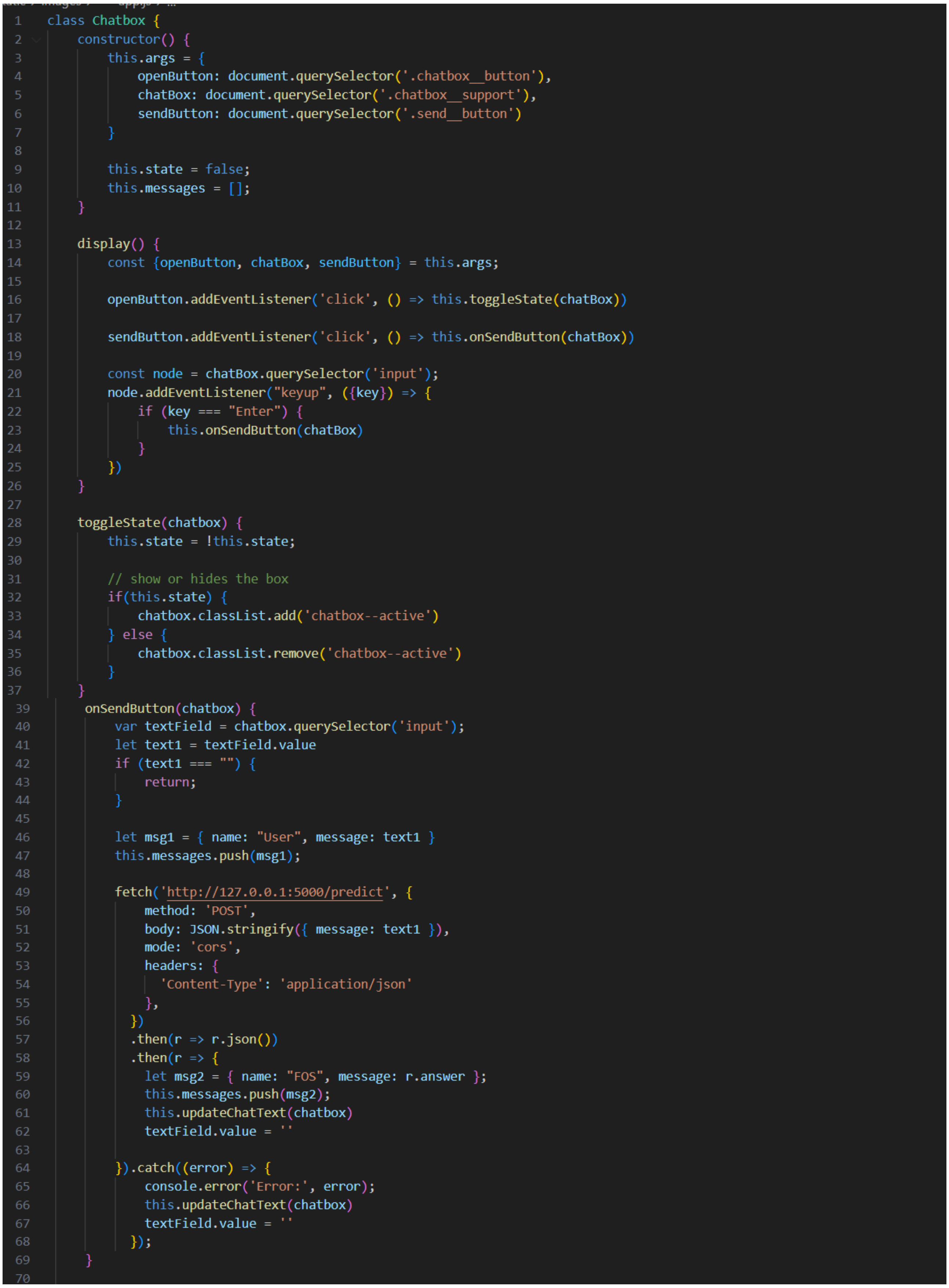


```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="style.css">

<head>
    <meta charset="UTF-8">
    <title>Chatbot</title>
</head>
<body>
<div class="container">
    <div class="chatbox">
        <div class="chatbox__support">
            <div class="chatbox__header">
                <div class="chatbox__image--header">
                    <img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image">
                </div>
                <div class="chatbox__content--header">
                    <h4 class="chatbox__heading--header">FOS BOT</h4>
                    <p class="chatbox__description--header">Hello. My name is FOS. How may I help you?</p>
                </div>
            </div>
            <div class="chatbox__messages">
                <div></div>
            </div>
            <div class="chatbox__footer">
                <input type="text" placeholder="Write a message...">
                <button class="chatbox__send--footer send__button">Send</button>
            </div>
        </div>
        <div class="chatbox__button">
            <button><img src="./images/chatbox-icon.svg" /></button>
        </div>
    </div>
</div>
```

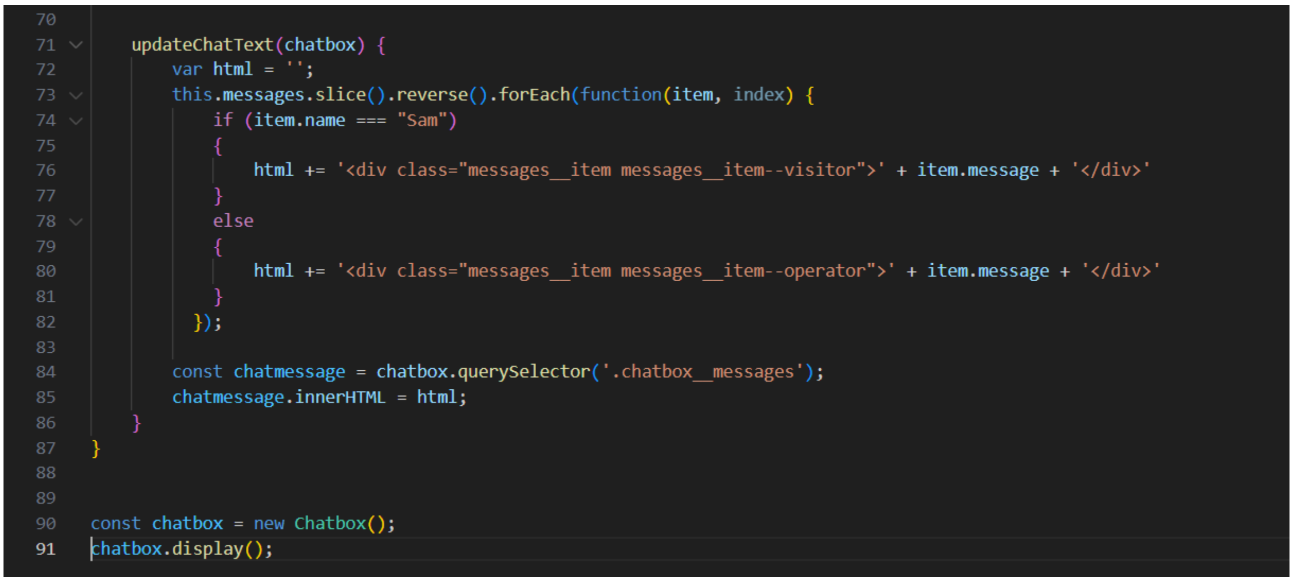


```html

    <script src="./app.js"></script>

</body>
</html>
```

Javascript Code:

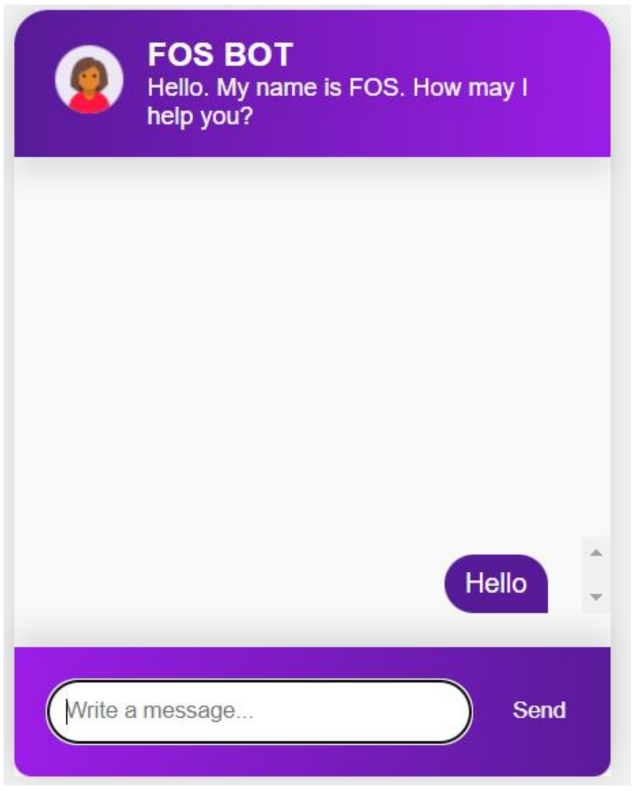


```javascript
class Chatbox {
    constructor() {
        this.args = {
            openButton: document.querySelector('.chatbox__button'),
            chatBox: document.querySelector('.chatbox__support'),
            sendButton: document.querySelector('.send__button')
        }

        this.state = false;
        this.messages = [];
    }

    display() {
        const {openButton, chatBox, sendButton} = this.args;

        openButton.addEventListener('click', () => this.toggleState(chatBox))

        sendButton.addEventListener('click', () => this.onSendButton(chatBox))

        const node = chatBox.querySelector('input');
        node.addEventListener("keyup", ({key}) => {
            if (key === "Enter") {
                this.onSendButton(chatBox)
            }
        })
    }

    toggleState(chatbox) {
        this.state = !this.state;

        // show or hides the box
        if(this.state) {
            chatbox.classList.add('chatbox--active')
        } else {
            chatbox.classList.remove('chatbox--active')
        }
    }
    onSendButton(chatbox) {
        var textField = chatbox.querySelector('input');
        let text1 = textField.value
        if (text1 === "") {
            return;
        }

        let msg1 = { name: "User", message: text1 }
        this.messages.push(msg1);

        fetch('http://127.0.0.1:5000/predict', {
            method: 'POST',
            body: JSON.stringify({ message: text1 }),
            mode: 'cors',
            headers: {
              'Content-Type': 'application/json'
            },
          })
          .then(r => r.json())
          .then(r => {
            let msg2 = { name: "FOS", message: r.answer };
            this.messages.push(msg2);
            this.updateChatText(chatbox)
            textField.value = ''

        }).catch((error) => {
            console.error('Error:', error);
            this.updateChatText(chatbox)
            textField.value = ''
          });
    }
```



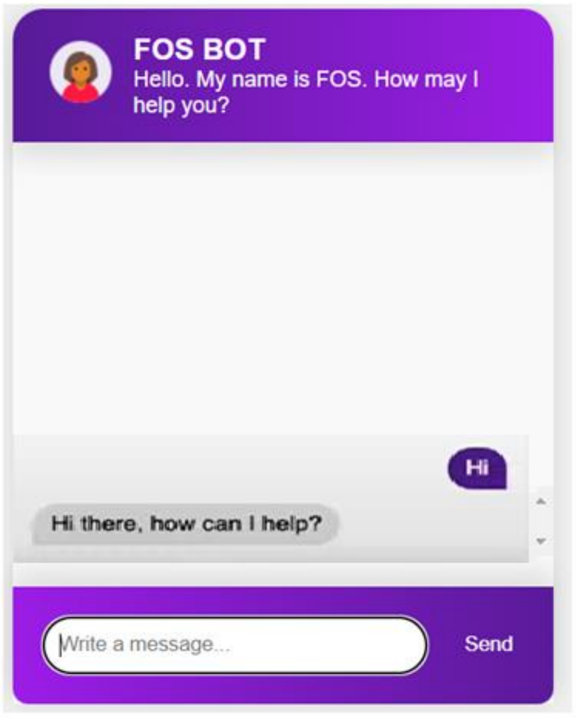
```
    updateChatText(chatbox) {
        var html = '';
        this.messages.slice().reverse().forEach(function(item, index) {
            if (item.name === "Sam")
            {
                html += '<div class="messages__item messages__item--visitor">' + item.message + '</div>'
            }
            else
            {
                html += '<div class="messages__item messages__item--operator">' + item.message + '</div>'
            }
          });

        const chatmessage = chatbox.querySelector('.chatbox__messages');
        chatmessage.innerHTML = html;
    }
}


const chatbox = new Chatbox();
chatbox.display();
```

OUTPUT:


FOS BOT
Hello. My name is FOS. How may I help you?

Hello

Write a message...
Send


FOS BOT
Hello. My name is FOS. How may I help you?

Hi

Hi there, how can I help?

Write a message...
Send

CHAPTER 6

INSTALLATION INSTRUCTIONS

To create this chatbot , one has to learn the basic natural processing techniques and Flask Framework.

Flask : pip install Flask

Transformers: pip install transformers

Openai: pip install openai

Speech Recognition: pip install SpeechRecognition

Text to speech: pip install pyttsx3

Flask cors: pip install flask-cors

Torchvision: pip install torchvision

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

In this work, we introduce FOS-BOT, a prototype chatbot that helps visually impaired people comprehend compositions or any other kind of information. The system can be thought of as an environment for information collection and probing that includes NLP-based features. FOS-BOT is a Natural Language Processing based AI-CHAT BOT. It takes document as an input which it should process according to the user’s instructions. Speech to text commands include paraphraser, inverted indexing, summarizer and query retrieval which can be given by the user. Correct Output is given as text to speech. The goal of this study is to determine whether and to what extent current NLP techniques can make it easier for students to acquire data that is relevant to their research. These methods provide quicker access to pertinent locations within a document as well as more information about the content of the document.

Future Enhancements:

- ➢ The project is to make a Chat-bot that will help visually impaired students with summarizing , paraphrasing , indexing and query retrieval of any information which they can upload as a document or directly put the information. Output will be text to speech.
- ➢ Speech recognition can be enhanced.
- ➢ Further enhancements to the user interface.
- ➢ The system is now evaluated more qualitatively than formally. Future work will undoubtedly include a formal assessment of the system with a variety of students to compare how they used it. Here, we want to make sure that we haven't created a system that caters to the tastes of a single person, but that it truly serves the majority of blind students. We intend to solicit the cooperation of a number of blind and non-blind students from a post-secondary institution for this. We do not believe that a fully quantitative review can be done, though. The apparent benefits might not correspond to the actual benefits, as is the case with all software. Overall, the system may be deemed useful if the user voluntarily decides to use it.

REFERENCES:

- Android based educational Chatbot for visually impaired people

- AI-based Chatbot for Physically Challenged People

- CSUN’s 19th Annual International Conference “Technology and Persons with Dis- abilities”, Los Angeles, March 2004.

- Julia Hodges, Shiyun Yie, Ray Reighart, and Lois Boggess. An automated system that assits in the generation of document indexes. Natural Language Engineering, 2:137–160, 1996.

- Christopher D. Manning and Hinrich Schu ̇tze. Foundations of Statistical Natural Language Processing. MIT Press, 1999.