

# **Text-to-speech & Voice-to-text Application**

**A Project Work**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**CSE - IBM Big Data Analytics**

**Submitted by:**

**Appana Venkata Naga Vineeth Sai - 18BCS3788**

**Ashish Mehra -18BCS3796**

**Divyanshu-18BCS3781**

**Akshat Sharma -18BCS3786**

**Under the Supervision of:**

**Ms. Jothi Pruthi**



**CHANDIGARH  
UNIVERSITY**  
Discover. Learn. Empower.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**APEX INSTITUTE OF TECHNOLOGY**

**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,  
PUNJAB**

**APRIL & 2018**

## Table of Contents

	Title Page	<i>I</i>
	Table of contents	<i>II</i>
	Declaration of the Student	<i>III</i>
	Abstract	<i>IV</i>
	Acknowledgement	<i>V</i>
	List of Figures	<i>VII- IX</i>
<b>1</b>	<b>INTRODUCTION</b>  1.1 Problem Definition 1.2 Project Overview/Specifications (page-1 and 3) 1.3 Hardware Specification 1.4 Software Specification	<b>1-2</b>  3 4 5 5
<b>2</b>	<b>LITERATURE SURVEY</b>  2.1 Literature review summary	6-59  60- 62
<b>3</b>	<b>PROBLEM FORMULATION</b>	63
<b>4</b>	<b>OBJECTIVES</b>	64
<b>5</b>	<b>METHODOLOGY</b>	65-88
<b>6</b>	<b>CONCLUSIONS AND DISCUSSION</b>	89
<b>7</b>	<b>REFERENCES</b>	90-92

*Annexure-2*

**DECLARATION**

I, ‘Appana Venkata Naga Vineethsai , Ashish Mehra, Akshat Sharma, Divyanshu’, student of ‘**Bachelor of Engineering in computer Science**’, session: **2018 – 2022**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled ‘**Text-To-Speech & Voice-To-Text Application**’ is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

(AVN.Vineethsai)

**Candidate UID: 18BCS3788**

(Divyanshu)

**Candidate UID: 18BCS3781**

(Ashish Mehra)

**Candidate UID: 18BCS3796**

(Akshat Sharma)

**Candidate UID: 18BCS3786**

**Date: 19-04-2022**

**Place: Chandigarh.**

## **ABSTRACT**

In present world, communication is the key element to progress. Passing on information, to the right person, and in the right manner is very important, not just on a corporate level, but also on a personal level. The world is moving towards digitization, so are the means of communication. Phone calls, emails, text messages etc. have become an integral part of message conveyance in this tech-savvy world. In order to serve the purpose of effective communication between two parties without hindrances, many applications have come to picture, which acts as a mediator and help in effectively carrying messages in form of text, or speech signals over miles of networks. Most of these applications find the use of functions such as articulatory and acoustic-based speech recognition, conversion from speech signals to text, and from text to synthetic speech signals, language translation amongst various others. In this review paper, we'll be observing different techniques and algorithms that are applied to achieve the mentioned functionalities.

**Keywords:** Voice to Text, Text to speech, Speech recognition, communication, Machine translation.

## **ACKNOWLEDGEMENT**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to Ms. Jyoti Pruthi, our project supervisor. We are highly indebted to Apex Institute of Technology, Chandigarh University for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our gratitude towards my parents & members of AIT for their kind cooperation and encouragement which helped me in completion of this project. We would like to express our special gratitude and thanks to industry persons for giving us such attention and time. Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

## **LIST OF FIGURES**

Table Title	page
<b>Figure 1:</b> Speech Recognition Process.....	7
<b>Figure 2:</b> flow chart input-MFCC.....	9
<b>Figure 3:</b> voice recognition system.....	13
<b>Figure 4:</b> speech synthesis.....	16
<b>Figure 5:</b> The time-frequency domain presentation of vowels.....	18
<b>Figure 6:</b> Algorithm of already existing systems.....	19
<b>Figure 7:</b> Phases of TTS synthesis process.....	20
<b>Figure 8:</b> Data flow diagram of the Speech synthesis system Using ..... Gane and Sarson Symbol.	21
<b>Figure 9:</b> High Level Model of the Proposed System.....	22
<b>Figure 10:</b> Flowchart representation of the program.....	23

<b>Figure 11:</b> overall view of the new system.....	29
<b>Figure 12:</b> Block diagram of text-to-speech device.....	30
<b>Figure 13:</b> Flowchart of OCR system .....	32
<b>Figure 14:</b> Software design of image processing module.....	33
<b>Figure 15:</b> A simple but general functional diagram of a TTS system.....	36
<b>Figure 16:</b> Operations of the natural Language processing module of a TTS synthesizer.....	42
<b>Figure 17:</b> The DSP component of a general concatenation based synthesizer.....	43
<b>Figure 18:</b> STT using HMM.....	46
<b>Figure 19:</b> TTS using HMM.....	48
<b>Figure 20:</b> Block diagram of speech synthesis.....	51
<b>Figure 21:</b> Basic structure of cascade formant synthesizer.....	56
<b>Figure 22:</b> Basic structure of a parallel formant synthesizer.....	57
<b>Figure 23:</b> API uses.....	65

<b>Figure 24:</b> TTS System Flow.....	66
<b>Figure 25:</b> Basic structure of cascade formant synthesizer.....	67
<b>Figure 26:</b> Basic structure of a parallel formant synthesizer.....	68
<b>Figure 27:</b> shows the voice working.....	69
<b>Figure 28:</b> shows the speak-set rate&pitch.....	70
<b>Figure 29:</b> coding part of html-1.....	72
<b>Figure 30:</b> coding part of html-2.....	73
<b>Figure 31:</b> coding part of js-1.....	74
<b>Figure 32:</b> coding part of js-2.....	75
<b>Figure 33:</b> coding part of js-3.....	76
<b>Figure 34:</b> output of TTS.....	77
<b>Figure 35:</b> final output of TTS.....	78
<b>Figure 36:</b> voice to text.....	79
<b>Figure 37:</b> Speech Recognition steps.....	80
<b>Figure 38:</b> Speech recognition, voice, and grammar object.....	82



<b>Figure 39:</b> coding part of html-1 .....	83
<b>Figure 40:</b> coding part of html-2 .....	84
<b>Figure 41:</b> coding part of js-1 .....	85
<b>Figure 42:</b> coding part of js-2 .....	86
<b>Figure 43:</b> allow microphone .....	87
<b>Figure 44:</b> copying text .....	87
<b>Figure 45:</b> final output of VTT .....	88

# 1 INTRODUCTION

Over the past few years, Mobile Phones have become an indispensable source of communication for the modern society. We can make calls and text messages from a source to a destination easily. It is known that verbal communication is the most appropriate medium of passing on and conceiving the correct information, avoiding misquotations. To fulfil the gap over a long distance, verbal communication can take place easily on phone calls. A path-breaking innovation has recently come to play in the SMS technology using the speech recognition technology, where voice messages are being converted to text messages. Quite a few applications used to assist the disabled make use of TTS, VTT, and translation. They can also be used for other applications, taking an example: Siri an intelligent automated assistant implemented on an electronic device, to facilitate user interaction with a device, and to help the user more effectively engage with local and/or remote services makes use of Nuance Communications voice recognition and text-to-speech (TTS) technology. In this paper, we will take a look at the different types of speech, speech recognition, speech to text conversion, text to speech conversion and speech translation. Under speech the recognition we will follow the method i.e. pre-emphasis of signals, feature extraction and recognition of the signals which help us in training and testing mechanism. There are various models used for this purpose but Dynamic time warp, which is used for feature extraction and distance measurement between features of signals and Hidden Markov Model which is a stochastic model and is used to connect various states of transition with each other is majorly used. Similarly for conversion of speech to text we use DTW and HMM models, along with various Neural Network models since they work well with phoneme classification, isolated word recognition, and speaker adaptation. End to end ASR is also being tested as of late

2014 to achieve similar results. Speech synthesis works well in helping convert tokenized words to artificial human speech. Different machine translation methods, as well as engines will also be reviewed and compared in this project. Following are the components of speech production, which are looked up to while applications use different speech related functionalities.

- Phonation (producing sound)
- Fluency
- Intonation
- Pitch variance
- Voice

In this project we have use Web Server API for the conversion of text-to-speech and from voice-to-speech. While using Web Server API it uses SpeechSynthesis (text-to-speech) and VoiceRecognition (voice-to-text) using Polymer. The Web Speech API has a main controller interface for this — SpeechRecognition — plus a number of closely-related interfaces for representing grammar, results, etc. Generally, the default speech recognition system available on the device will be used for the speech recognition — most modern OSes have a speech recognition system for issuing voice commands.

## **1.1 Problem Definition**

Text-to-speech & Voice-to-text is very important nowadays. It can be used in many applications. Speech is one of the oldest and most natural means of information exchange between human. Over the years, Attempts have been made to develop vocally interactive computers to realise voice/speech synthesis. Obviously such an interface would yield great benefits. In this case a computer can synthesize text and give out a speech. Text-To-Speech Synthesis is a Technology that provides a means of converting written text from a descriptive form to a spoken language that is easily understandable by the end user (Basically in English Language). In case of Voice-to-text, many of us are living in a very busy world where Multi-tasking is the order of the day. Many people we saw taking notes on phone and performing some other random activities at the same time, so in this case the voice-to-text technology helpful for many people. So, In this project we are trying to explore potential solutions to create one application were anyone can easily convert Text-to-speech & Voice-to-text.

## 1.2 Project Overview

In this project we are working with API (Application Programming Interface) is an interface that defines interactions between multiple Software applications. We divide the some production types like:

- Phonation (sound)
- Fluency
- Intonation
- Pitch variance
- Voice

In Text-To-Speech we are using Web API, and Speech Synthesis API, to convert Texts to speech in different voices. And in Voice-to-text we using Speech Recognition Web API, another component of the Web Speech API that is used to convert Voice to Texts, and then build a Web application that converts our words/sentences to text format. Coming to project result we are not preparing any working product, we decided to do application based project that will help to convert Text-to-speech & Voice-to-text.

Finally the output from the project is that application based project, research paper, report and ppt.

### **1.3 Hardware Specification**

Product name- HP Pavilion x360

Processor family - Intel(R) Core(TM) i7

Memory - 8 GB DDR4-2400 SDRAM (1 x 8 GB)

### **1.4 Software Specification**

Brackets project: Brackets is a text editor that can be used for web development & application based projects. It is designed to help for us developers and designers in coding CSS, HTML, and Javascript easier and faster.

PROGRAMMING LANGUAGES: HTML, CSS, JAVASCRIPT

## 2 LITERATURE REVIEW

### 2.0.1 Speech Recognition

Speech Recognition is the ability of machine/program to identify words and phrases in spoken language and convert them into machine-readable format. Speech Recognition Systems can be classified on basis of the following parameters:

- **Speaker:** All speakers have a different kind of voice. The models hence are either designed for a specific speaker or an independent speaker.
- **Vocal Sound:** The way the speaker speaks also plays a role in speech recognition. Some models can recognize either single utterances or separate utterance with a pause in between.
- **Vocabulary:** The size of the vocabulary plays an important role in determining the complexity, performance, and precision of the system.

Basic Speech Recognition Model: Each speech recognition system follow some standard steps as shown in figure 1.

- i) **Pre-processing:** The analog speech signal is transformed into digital signals for later processing. This digital signal is moved to the first order filters to spectrally flatten the signals. This helps in increasing the signal's energy at a higher frequency.

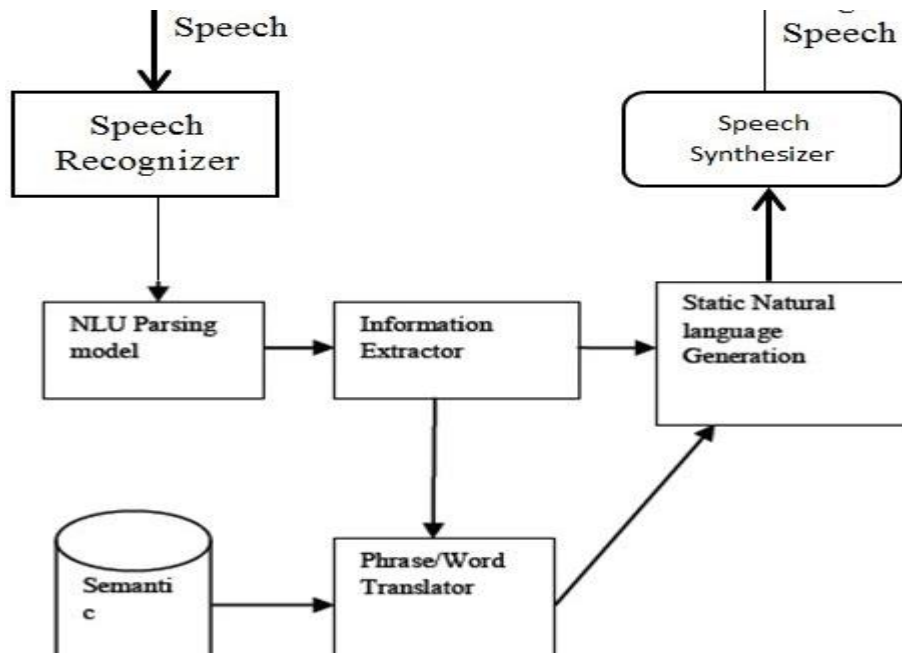


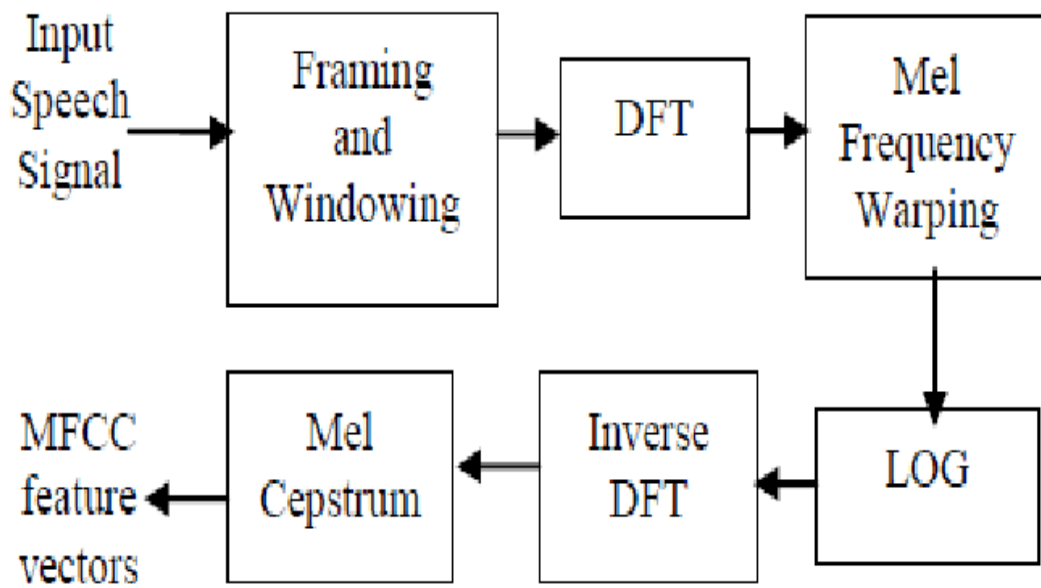
Fig.1.SpeechRecognitionProcess

ii) **Feature Extraction:** This step finds the set of parameters of utterances that have a correlation with speech signals. These parameters, known as features, are computed through processing of the acoustic waveform. The main focus is to compute a sequence of feature vectors (relevant information) providing a compact representation of the given input signal. Commonly used feature extraction techniques are discussed below:

- **Linear Predictive Coding (LPC):** The basic idea is that the speech sample can be approximated as a linear combination of past speech samples. Figure 2 shows the LPC process [2]. The digitized signal is blocked into frames of N samples. Then each sample frame is windowed to minimize signal discontinuities. Each framed window is then auto-correlated. The last step is the LPC analysis, which converts each frame of autocorrelations into LPC parameter set.



- **Mel-Frequency Cestrum Co-efficient (MFCC):** It is a very powerful technique and uses human auditory perception system. MFCC applies certain steps to the input signal: Framing: Speech wave- form is cropped to remove interference if present; Windowing: minimizes the discontinuities in the signal; Discrete Fourier Transform: converts each frame from time domain to frequency domain; Mel Filter Bank Algorithm: the signal is plotted against the Mel spectrum to mimic human hearing.
  
  - **Dynamic Time Warping:** This algorithm is used for measuring the similarity between two-time series which may vary in speed, based on dynamic programming. It aims at aligning two sequences of feature vectors (1 of each series) iteratively until an optimal match (according to a suitable metrics) between them is found.
- iii) **Acoustic Models:** It is the fundamental part of Automated Speech Recognition (ASR) system where a connection between the acoustic information and phonetics is established. Training establishes a correlation between the basic speech units and the acoustic observations.



**Fig.2.** flow chart input-MFCC

- iv) **Language Models:** This model induces the probability of a word occurrence after a word sequence. It contains the structural constraints available in the language to generate the probabilities of occurrence. The language model distinguishes word and phrase that has a similar sound.
  
- v) **Pattern Classification:** It is the process of comparing the unknown pattern with existing sound reference pattern and computing similarity between them. After completing the training of the system at the time of testing, patterns are classified to recognize the speech. Different approaches for pattern matching are :

- **Template Based Approach:** This approach has a collection of speech patterns which are stored as a reference representing dictionary words. Speech is recognized by matching the uttered word with the reference template [14].
- **Knowledge Based Approach:** This approach takes set of features from the speech and then train the system to generate set of production rules automatically from the samples.
- **Neural Network Based Approach:** This approach is capable of solving more complicated recognition task. The basic idea is to compile and incorporate knowledge from a variety of knowledge sources with the problem at hand [8].
- **Statistical Based Approach:** In this approach, variations in speech are modelled statistically (e.g. HMM) using training methods

## 2.0.2 Classification of speech recognition system:

Speech recognition system can be classified in several different types by describing the type of speech utterance, type of speaker model and type of vocabulary that they have the ability to recognize. The challenges are briefly explained below:

### A. Types of speech utterance

Speech recognition are classified according to what type of utterance they have ability to recognize. They are classified as:

- 1) **Isolated word:** Isolated word recognizer usually requires each spoken word to have quiet (lack of an audio signal) on both side of the sample window. It accepts single word at a time.
- 2) **Connected word:** It is similar to isolated word, but it allows separate utterances to „run-together“ which contains a minimum pause in between them.
- 3) **Continuous Speech:** it allows the users to speak naturally and in parallel the computer will determine the content.
- 4) **Spontaneous Speech:** It is the type of speech which is natural sounding and is not rehearsed.

## **B. Types of speaker model**

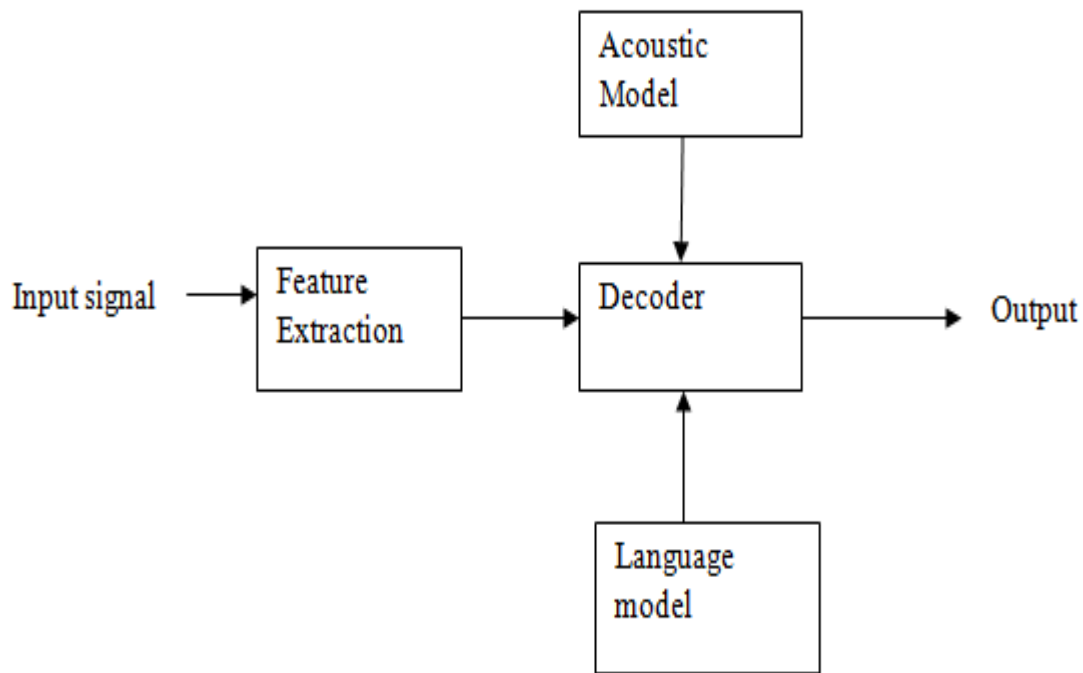
Speech recognition system is broadly into two main categories based on speaker models namely speaker dependent and speaker independent.

- 1) **Speaker dependent models:** These systems are designed for a specific speaker. They are easier to develop and more accurate but they are not so flexible.
- 2) **Speaker independent models:** These systems are designed for variety of speaker. These systems are difficult to develop and less accurate but they are very much flexible.

## **C. Types of vocabulary**

The vocabulary size of speech recognition system affects the processing requirements, accuracy and complexity of the system. In voice recognition system:speech-to-text the types of vocabularies can be classified as follows:

- 1) **Small vocabulary:** single letter.
- 2) **Medium vocabulary:** two or three letter words.
- 3) **Large vocabulary:** more letter words.



**Fig.3.**voice recognition system

Figure 1: Overview of Voice Recognition System:Speech-to-text.

**Input signal-** Voice input by the user.

**Feature Extraction:** it should retain useful information of the signal, deduct redundant and unwanted information, show less variation from one speaking environment to another, occur normally and naturally in speech.

**Acoustic model:** it contains statistical representations of each distinct sounds that makes up a word.

**Decoder:** it will decode the input signal after feature extraction and will show the desired output. Language model- it assigns a probability to a sequence of words by means of a probability distribution.

**Output-** interpreted text is given by the computer.

The main of the project is to recognize speech using MFCC and VQ techniques. The feature extraction will be done using Mel Frequency Cepstral Coefficients(MFCC). The steps of MFCC are as follows:-

- 1) Framing and Blocking
- 2) Windowing
- 3) FFT(Fast Fourier Transform)
- 4) Mel-Scale
- 5) Discrete Cosine Transform(DCT)

Feature matching will be done using Vector Quantization technique. The steps are as follows:-

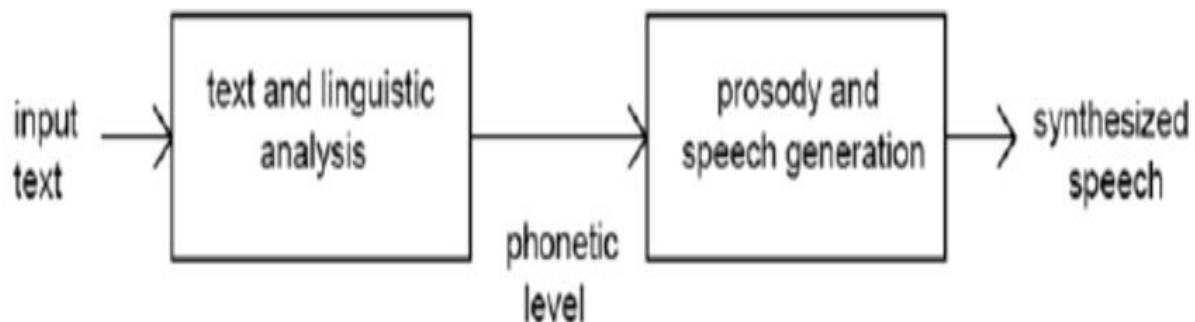
- 1) By choosing any two dimensions, inspection on vectors is done and data points are plotted.

- 2) To check whether data region for two different speaker are overlapping each other and in same cluster, observation is needed.
- 3) Using LGB algorithm Function Vqlbg will train the VQ codebook. The extracted features will be stored in .mat file using MFCC algorithm. Models will be created using Hidden Markov Model(HMM). The desired output will be shown in matlab interface.



### 2.0.3 Text – To - Speech Synthesis Defined

A speech synthesis system is by definition a system, which produces synthetic speech. It is implicitly clear, that this involves some sort of input. What is not clear is the type of this input. If the input is plain text, which does not contain additional phonetic and/or phonological information the system may be called a text-to-speech (TTS) system. A schematic of the text-to-speech process is shown in the figure 1 below. As shown, the synthesis starts from text input. Nowadays this may be plain text or marked-up text e.g. HTML or something similar like JSML (Java Synthesis Mark-up Language).

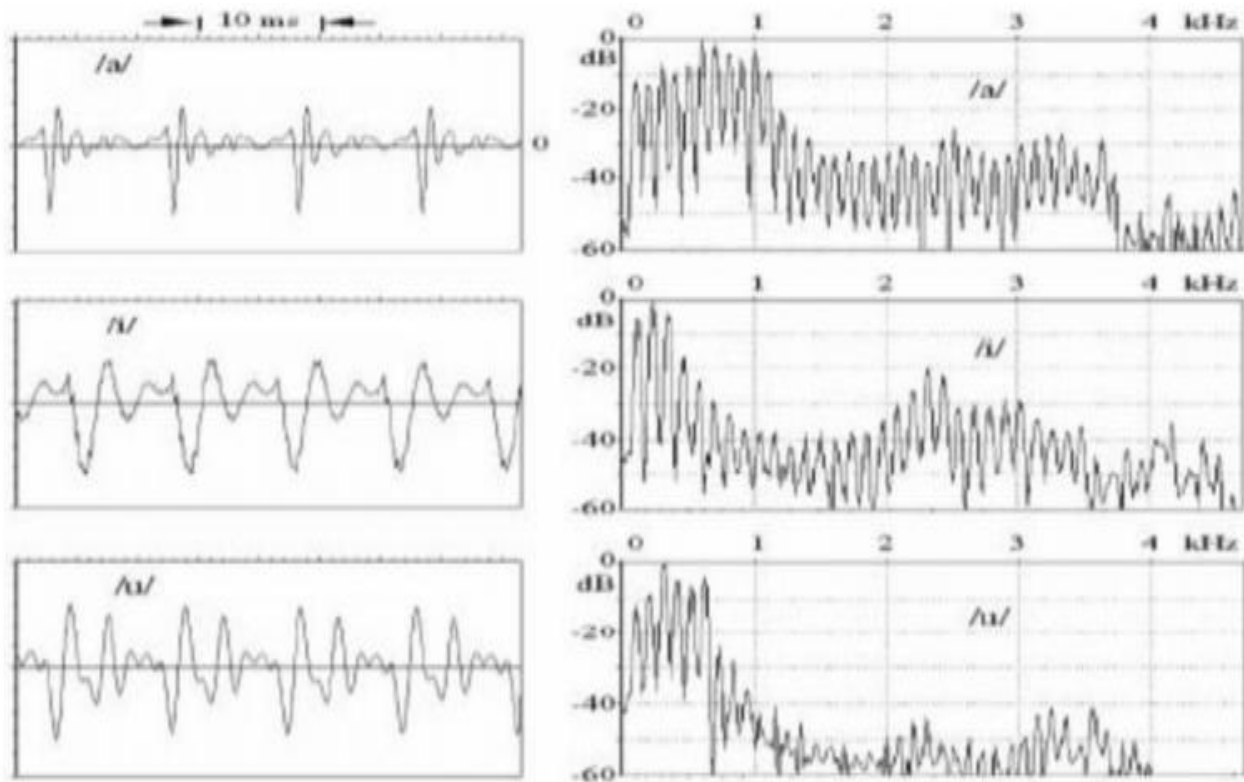


**Fig.4.**speech synthesis

### Representation and Analysis of Speech Signals

Continuous speech is a set of complicated audio signals which makes producing them artificially difficult. Speech signals are usually considered as voiced or

unvoiced, but in some cases they are something between these two. Voiced sounds consist of fundamental frequency ( $F_0$ ) and its harmonic components produced by vocal cords (vocal folds). The vocal tract modifies this excitation signal causing formant (pole) and sometimes anti-formant (zero) frequencies (Abedjieva et al., 1993). Each formant frequency has also amplitude and bandwidth and it may be sometimes difficult to define some of these parameters correctly. The fundamental frequency and formant frequencies are probably the most important concepts in speech synthesis and also in speech processing. With purely unvoiced sounds, there is no fundamental frequency in excitation signal and therefore no harmonic structure either and the excitation can be considered as white noise. The airflow is forced through a vocal tract constriction which can occur in several places between glottis and mouth. Some sounds are produced with complete stoppage of airflow followed by a sudden release, producing an impulsive turbulent excitation often followed by a more protracted turbulent excitation (Allen et al., 1987). Unvoiced sounds are also usually more silent and less steady than voiced ones. Speech signals of the three vowels (/a/ /i/ /u/) are presented in time-frequency domain in Figure 3. The fundamental frequency is about 100 Hz in all cases and the formant frequencies  $F_1$ ,  $F_2$ , and  $F_3$  with vowel /a/ are approximately 600 Hz, 1000 Hz, and 2500 Hz respectively. With vowel /i/ the first three formants are 200 Hz, 2300 Hz, and 3000 Hz, and with /u/ 300 Hz, 600 Hz, and 2300 Hz.



**Fig.5.** The time-frequency domain presentation of vowels /a/, /i/, and /u/.

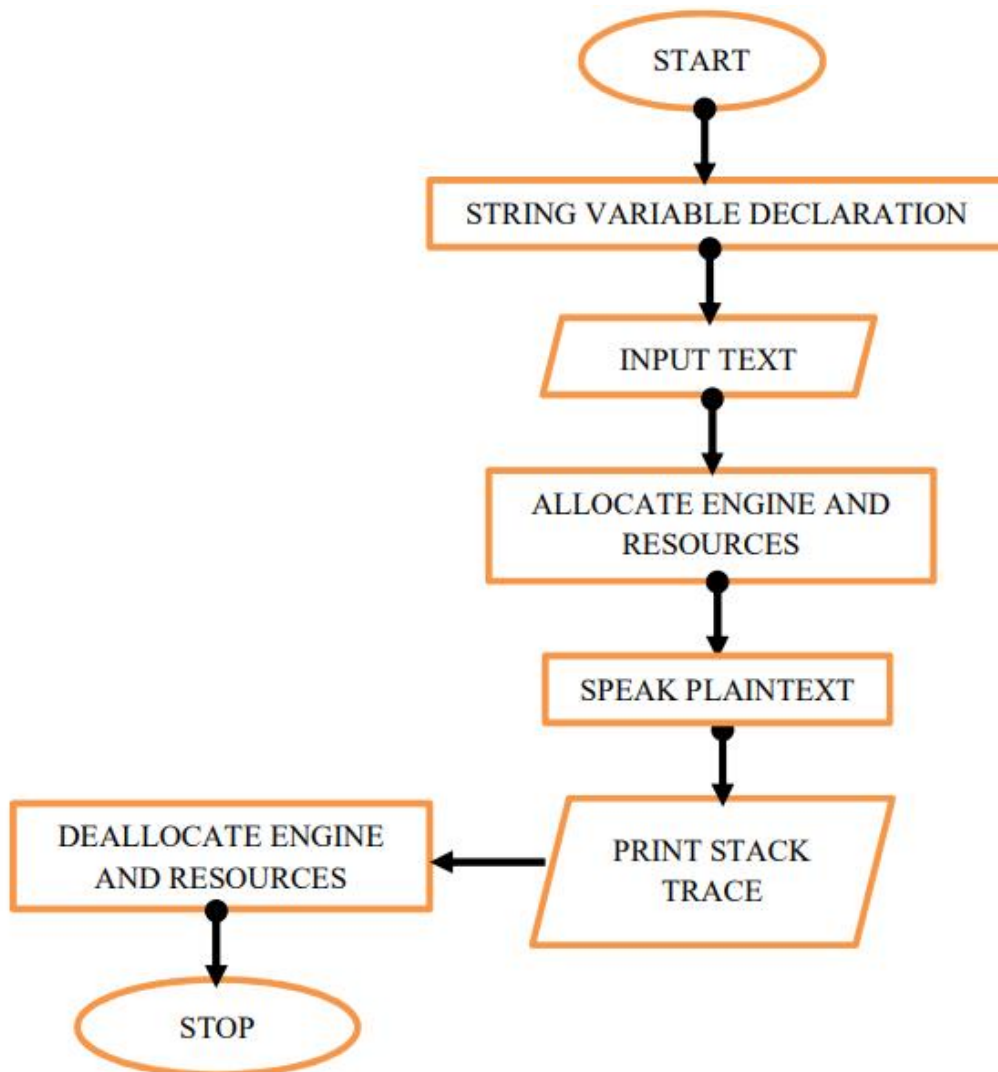
## Analysis and Problems of Existing Systems

Existing systems algorithm is shown below in Figure 6. It shows that the system does not have an avenue to annotate text to the specification of the user rather it speaks plaintext.

Due studies revealed the following inadequacies with already existing systems:

1. **Structure analysis:** punctuation and formatting do not indicate where paragraphs and other structures start and end. For example, the final period in “P.D.P.” might be misinterpreted as the end of a sentence.

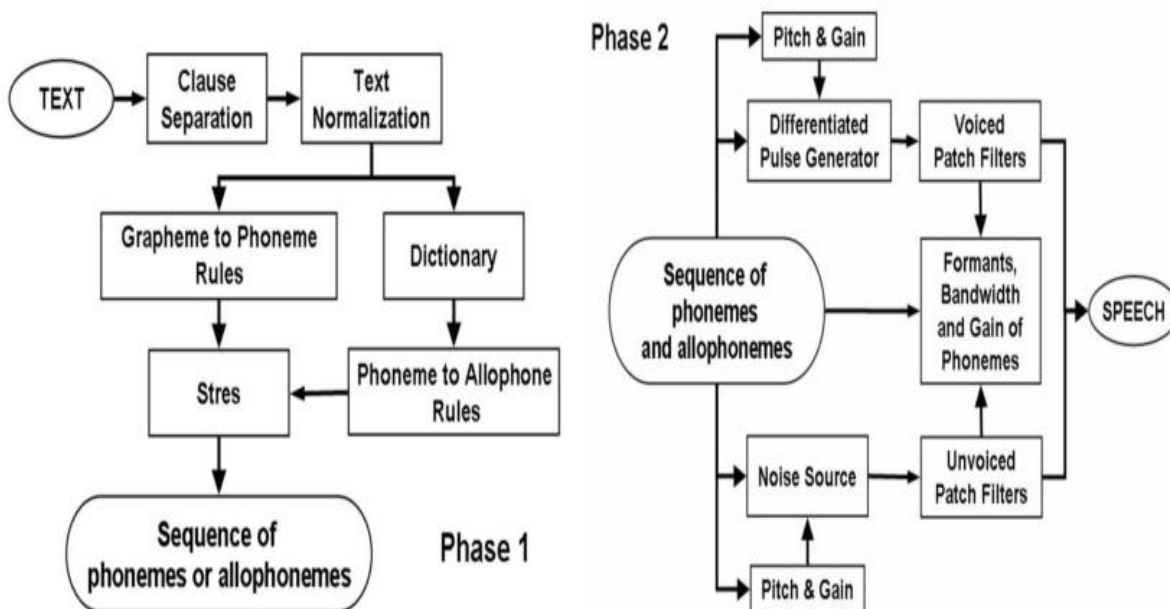
2. **Text pre-processing:** The system only produces the text that is fed into it without any pre-processing operation occurring.
3. **Text-to-phoneme conversion:** existing synthesizer system can pronounce tens of thousands or even hundreds of thousands of words correctly if the word(s) is/are not found in the data dictionary.



**Fig.6.** Algorithm of already existing systems

## Speech Synthesis Module

The TTS system converts an arbitrary ASCII text to speech. The first step involves extracting the phonetic components of the message, and we obtain a string of symbols representing sound-units (phonemes or allophones), boundaries between words, phrases and sentences along with a set of prosody markers (indicating the speed, the intonation etc.). The second step consists of finding the match between the sequence of symbols and appropriate items stored in the phonetic inventory and binding them together to form the acoustic signal for the voice output device.



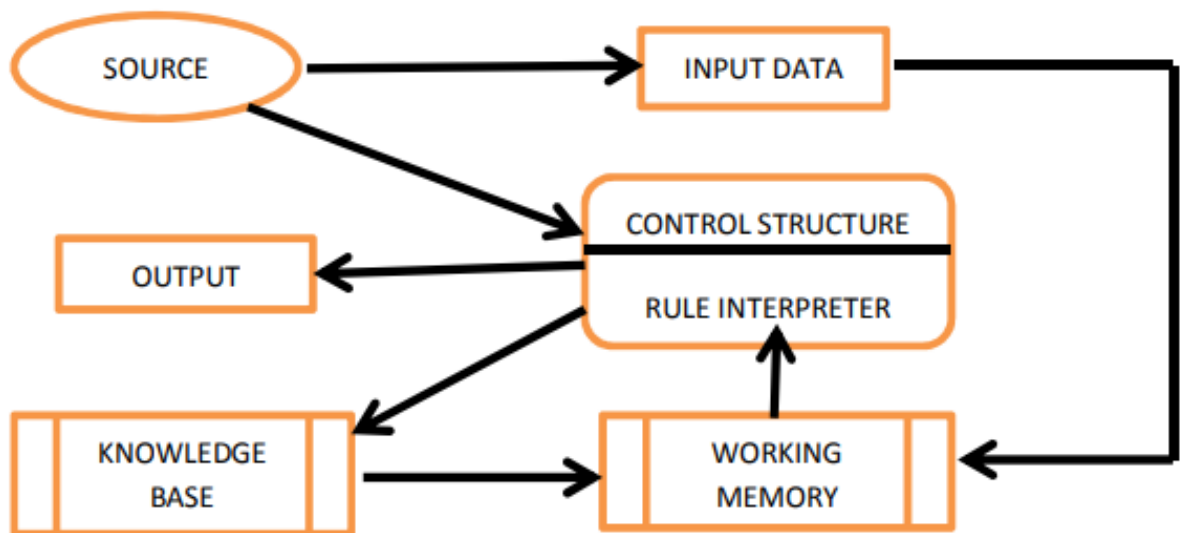
**Fig.7.** Phases of TTS synthesis process

To compute the output, the system consults

1. A database containing the parameter values for the sounds within the word,
2. A knowledge base enumerating the options for synthesizing the sounds.

Incorporating Expert system in the internal programs will enable the new TTS system exhibit these features:

1. The system performs at a level generally recognized as equivalent to that of a human expert
2. The system is highly domain specific.
3. The system can explain its reasoning process
4. If the information with which it is working is probabilistic or fuzzy, the system can correctly propagate uncertainties and provide a range of alternative solution with associated likelihood.



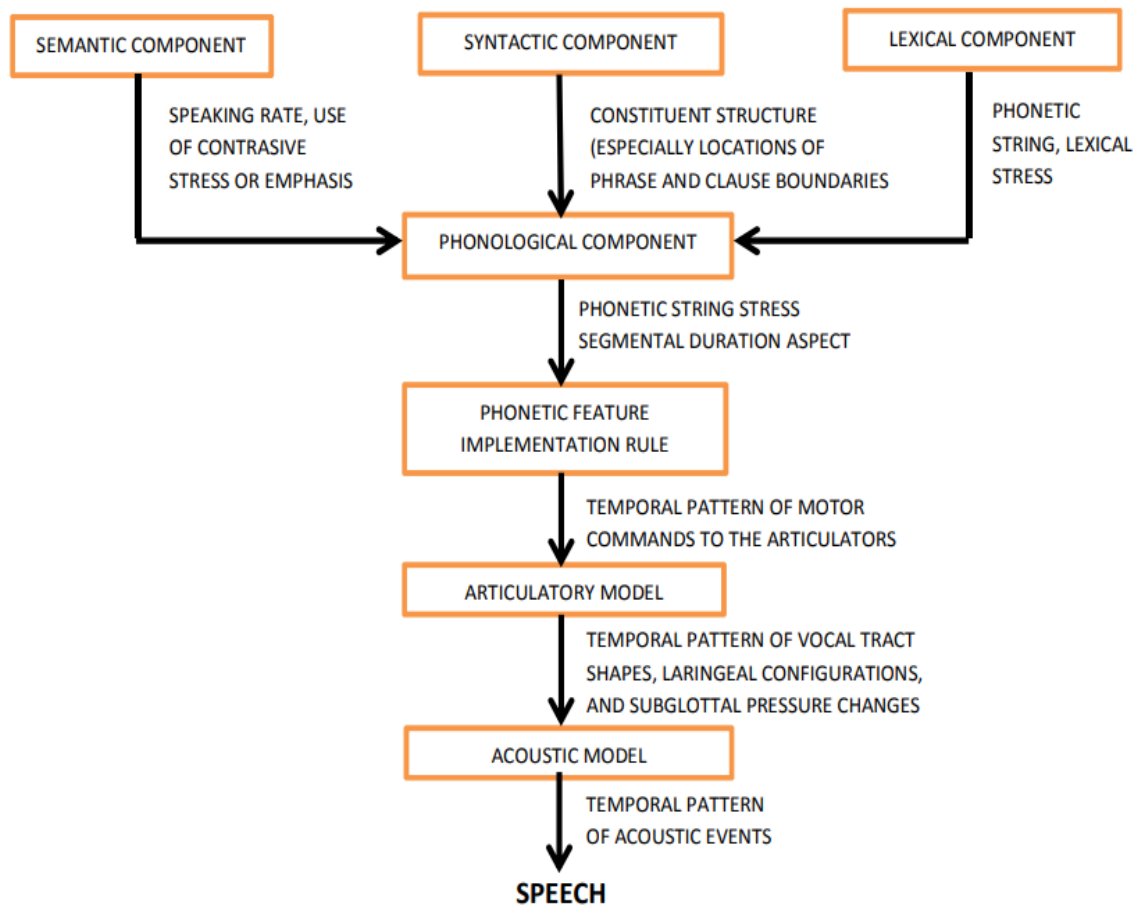
**Fig.8.** Data flow diagram of the Speech synthesis system Using Gane and Sarson Symbol

**User Interface (Source):** This can be Graphical User Interface (GUI), or the Command Line Interface (CLI).

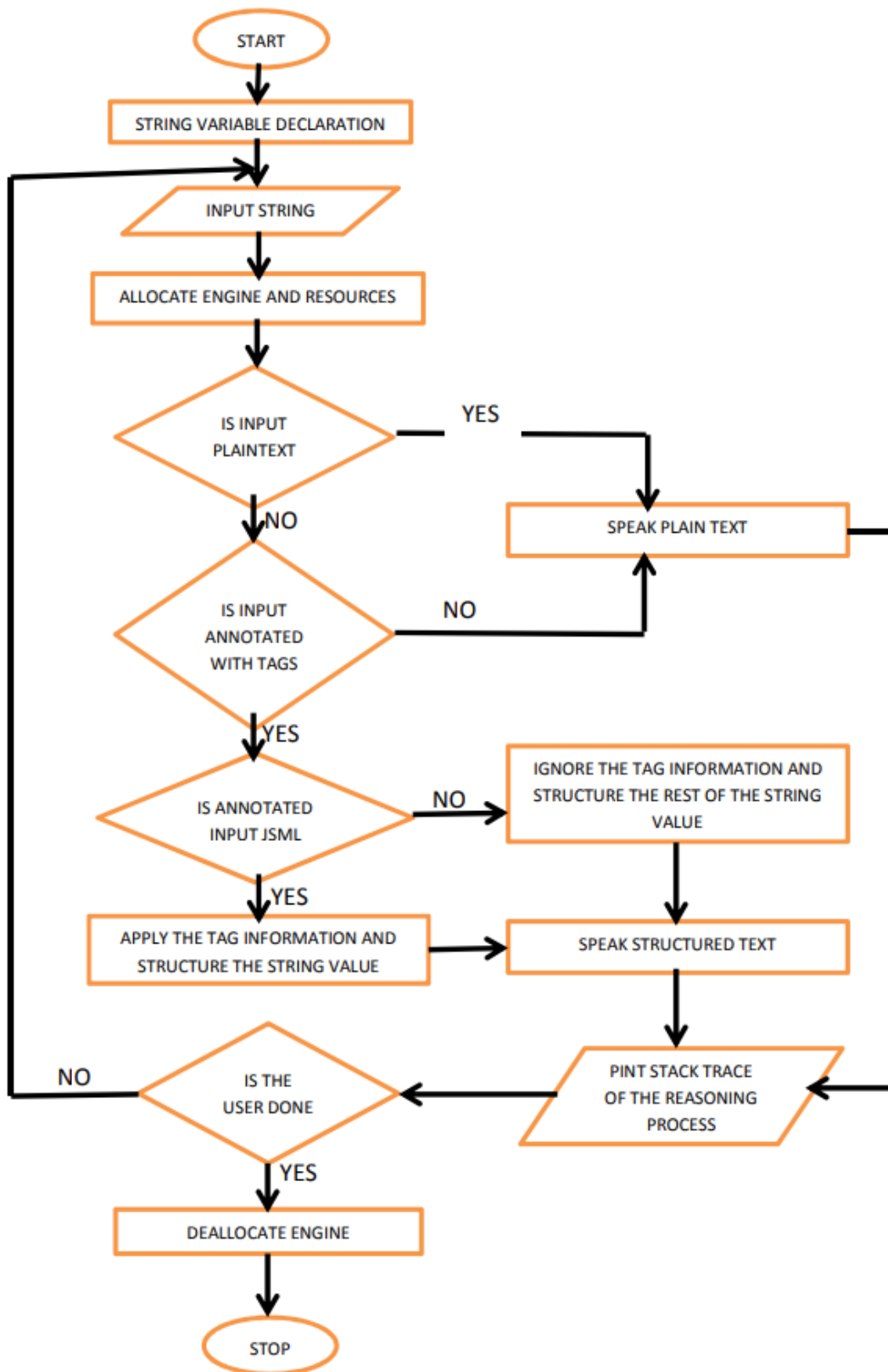
**Knowledge Base (Rule set):** FreeTTS module/system/engine. This source of the knowledge includes domain specific facts and heuristics useful for solving problems in the domain. FreeTTS is an open source speech synthesis system written entirely in the Java programming language. It is based upon Flite. FreeTTS is an implementation of Sun's Java Speech API. FreeTTS supports end-of-speech markers.

**Control Structures:** This rule interpreter inference engine applies to the knowledge base information for solving the problem.

**Short term memory:** The working memory registers the current problem status and history of solution to date.



**Fig.9.** High Level Model of the Proposed System



**Fig.10.** Flowchart representation of the program



## **Choice of Speech Engine and Programming Language**

The speech engine used in this new system was the FreeTTS speech engine. FreeTTS was used because it is programmed using JAVA (the backbone programming language of this designed TTS system). It also supports SAPI (Speech Application Programming Interface) which is in synchronism with the JSAPI (Java Speech Application Programming Interface). JSAPI was also the standardized interface used in the new system. FreeTTS includes an engine for the vocal synthesis that supports a certain number of voices (male and female) at different frequencies. It is recommended to use JSAPI to interface with FreeTTS because JSAPI interface provides the best methods of controlling and using FreeTTS. FreeTTS engine enable full control about the speech signal. This new designed system provides the possibility to choose a voice between three types of voices: an 8 kHz, diphone male English voice named kevin, a 16 kHz diphone male English voice named kevin16 and a16khz limited domain, male US English voice named alan. The user could also set the properties of a chosen voice: the speaking rate, the volume and the pitch. A determining factor in the choice of programming language is the special connotation (JSML) given to the program. This is a java specification mark-up language used to annotate spoken output to the preferred construct of the user. In addition to this, there is the need for a language that supports third party development of program libraries

for use in a particular situation that is not amongst the specification of the original platform. Considering these factors, the best choice of programming language was JAVA. Other factors that made JAVA suitable were its dual nature (i.e. implementing 2 methodologies with one language), its ability to Implements proper data hiding technique (Encapsulation), its supports for inner abstract class or object development, and its ability to provide the capability of polymorphism; which is a key property of the program in question

### **Design of the New System**

Some of the technologies involved in the design of this system includes the following: Speech Application Programming Interface (SAPI): SAPI is an interface between applications and speech technology engines, both text-to-speech and speech recognition (Amundsen 1996). The interface allows multiple applications to share the available speech resources on a computer without having to program the speech engine itself. SAPI consists of three interfaces; The voice text interface which provides methods to start, pause, resume, fast forward, rewind, and stop the TTS engine during speech. The attribute interface allows access to control the basic behaviour of the TTS engine. Finally, the dialog interface can be used to set and

retrieve information regarding the TTS engine. Java Speech API (JSAPI): The Java Speech API defines a standard, easy-to-use, cross-platform software interface to state-of-the-art speech technology. Two core speech technologies supported through the Java Speech API are speech recognition and speech synthesis. Speech recognition provides computers with the ability to listen to spoken language and to determine what has been said. Speech synthesis provides the reverse process of producing synthetic speech from text generated by an application, an applet or a user. It is often referred to as text-to-speech technology.

### **Choice of Speech Engine and Programming Language**

The speech engine used in this new system was the FreeTTS speech engine. FreeTTS was used because it is programmed using JAVA (the backbone programming language of this designed TTS system). It also supports SAPI (Speech Application Programming Interface) which is in synchronism with the JSAPI (Java Speech Application Programming Interface). JSAPI was also the standardized interface used in the new system. FreeTTS includes an engine for the vocal synthesis that supports a certain number of voices (male and female) at different frequencies. It is recommended to use JSAPI to interface with FreeTTS because JSAPI interface provides the best methods of controlling and using FreeTTS. FreeTTS engine enable full control about the speech signal. This new designed system provides the

possibility to choose a voice between three types of voices: an 8 kHz, diphone male English voice named kevin, a 16 kHz diphone male English voice named kevin16 and a16khz limited domain, male US English voice named alan. The user could also set the properties of a chosen voice: the speaking rate, the volume and the pitch.

A determining factor in the choice of programming language is the special connotation (JSML) given to the program. This is a java specification mark-up language used to annotate spoken output to the preferred construct of the user. In addition to this, there is the need for a language that supports third party development of program libraries for use in a particular situation that is not amongst the specification of the original platform. Considering these factors, the best choice of programming language was JAVA. Other factors that made JAVA suitable were its dual nature (i.e. implementing 2 methodologies with one language), its ability to Implements proper data hiding technique (Encapsulation), its supports for inner abstract class or object development, and its ability to provide the capability of polymorphism; which is a key property of the program in question.

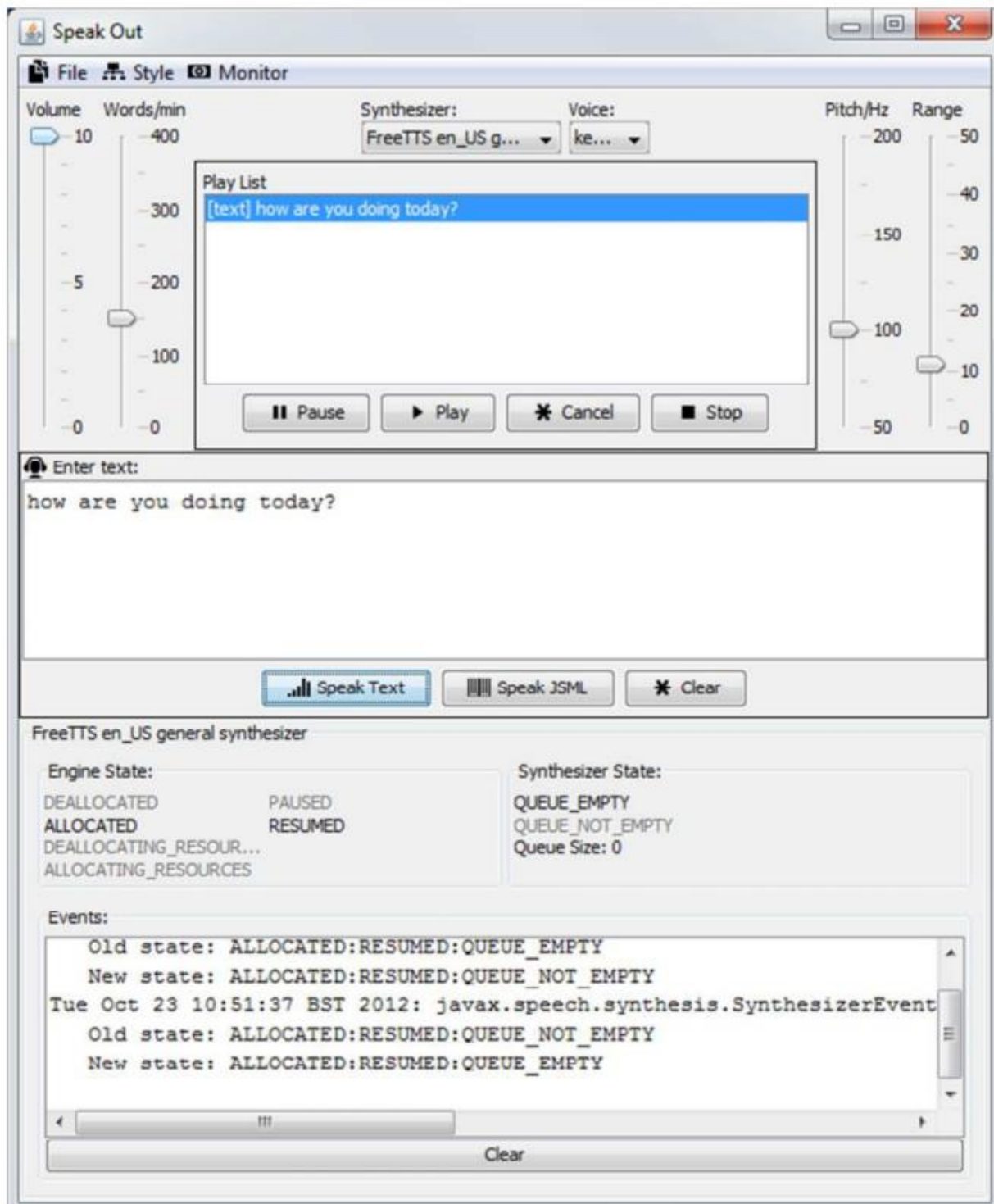
Some of the technologies involved in the design of this system includes the following:

## **Speech Application Programming Interface (SAPI):**

SAPI is an interface between applications and speech technology engines, both text-to-speech and speech recognition (Amundsen 1996). The interface allows multiple applications to share the available speech resources on a computer without having to program the speech engine itself. SAPI consists of three interfaces; The voice text interface which provides methods to start, pause, resume, fast forward, rewind, and stop the TTS engine during speech. The attribute interface allows access to control the basic behaviour of the TTS engine. Finally, the dialog interface can be used to set and retrieve information regarding the TTS engine.

## **Java Speech API (JSAPI):**

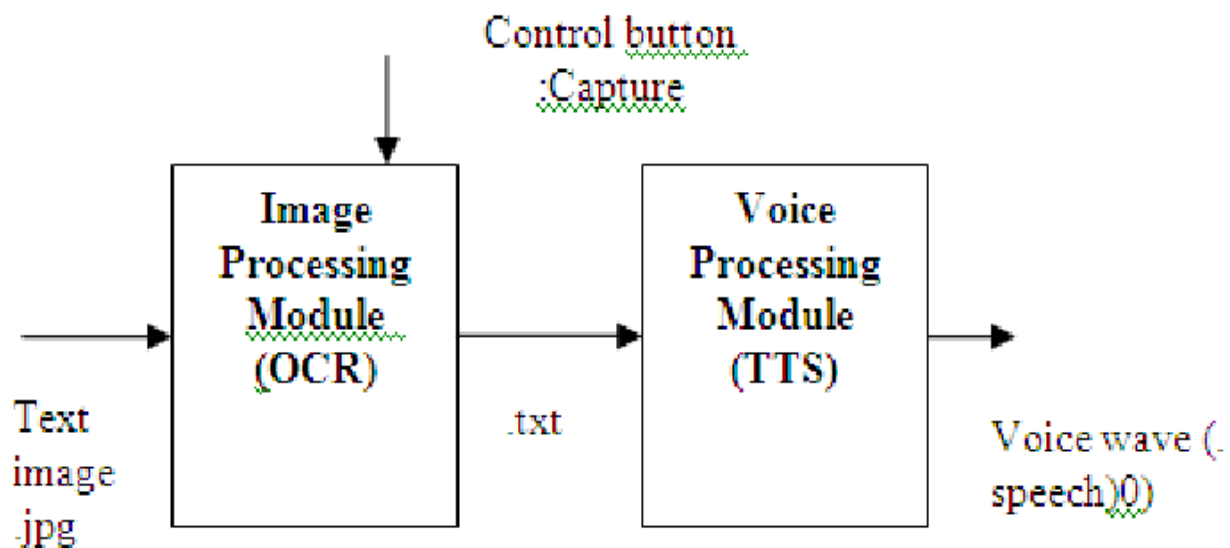
The Java Speech API defines a standard, easy-to-use, cross-platform software interface to state-of-the-art speech technology. Two core speech technologies supported through the Java Speech API are speech recognition and speech synthesis. Speech recognition provides computers with the ability to listen to spoken language and to determine what has been said. Speech synthesis provides the reverse process of producing synthetic speech from text generated by an application, an applet or a user. It is often referred to as text-to-speech technology.



**Fig.11.**overall view of the new system.

## 2.0.4 OCR Implementation

Text-to-speech device consists of two main modules, the image processing module and voice processing modules. Image processing module captures image using camera, converting the image into text. Voice processing module changes the text into sound and processes it with specific physical characteristics so that the sound can be understood. Figure 2 shows the block diagram of TextTo-Speech device, 1st block is image processing module, where OCR converts .jpg to .txt form. 2nd is voice processing module which converts .txt to speech

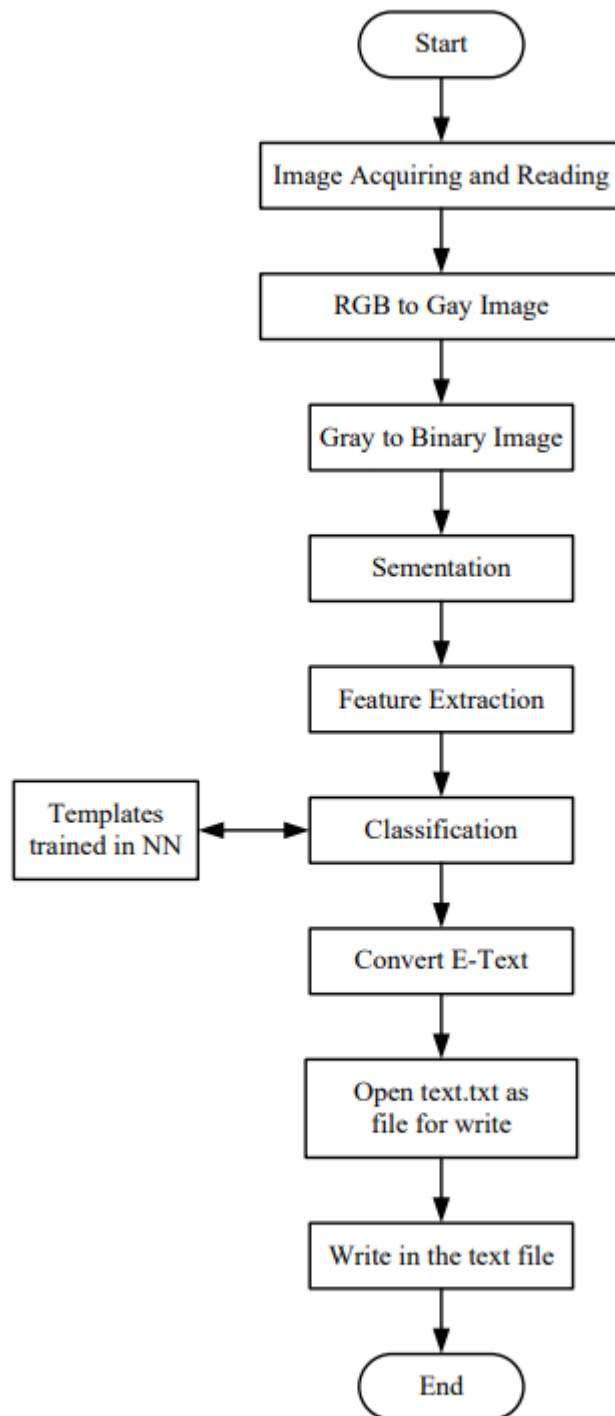


**Fig.12.** Block diagram of text-to-speech device.

Figure 11 shows the block diagram of Text-To-Speech device, 1st block is image processing module, where OCR converts .jpg to .txt form. 2nd is voice processing module which converts .txt to speech. OCR is important element in this module. OCR or Optical Character Recognition is a technology that automatically recognize the character through the optical mechanism, this technology imitate the ability of the human senses of sight, where the camera becomes a replacement for eye and image processing is done in the computer engine as a substitute for the human brain<sup>2</sup> . Tesseract OCR is a type of OCR engine with matrix matching<sup>3</sup> . The selection of Tesseract engine is because of its flexibility and extensibility of machines and the fact that many communities are active researchers to develop this OCR engine and also because Tesseract OCR can support 149 languages. In this project we are identifying English alphabets. Before feeding the image to the OCR, it is converted to a binary image to increase the recognition accuracy. Image binary conversion is done by using Imagemagick software, which is another open source tool for image manipulation. The output of OCR is the text, which is stored in a file (speech.txt). Machines still have defects such as distortion at the edges and dim light effect, so it is still difficult for most OCR engines to get high accuracy text<sup>4</sup> . It needs some supporting and condition in order to get the minimal defect. Tesseract OCR Implementation.



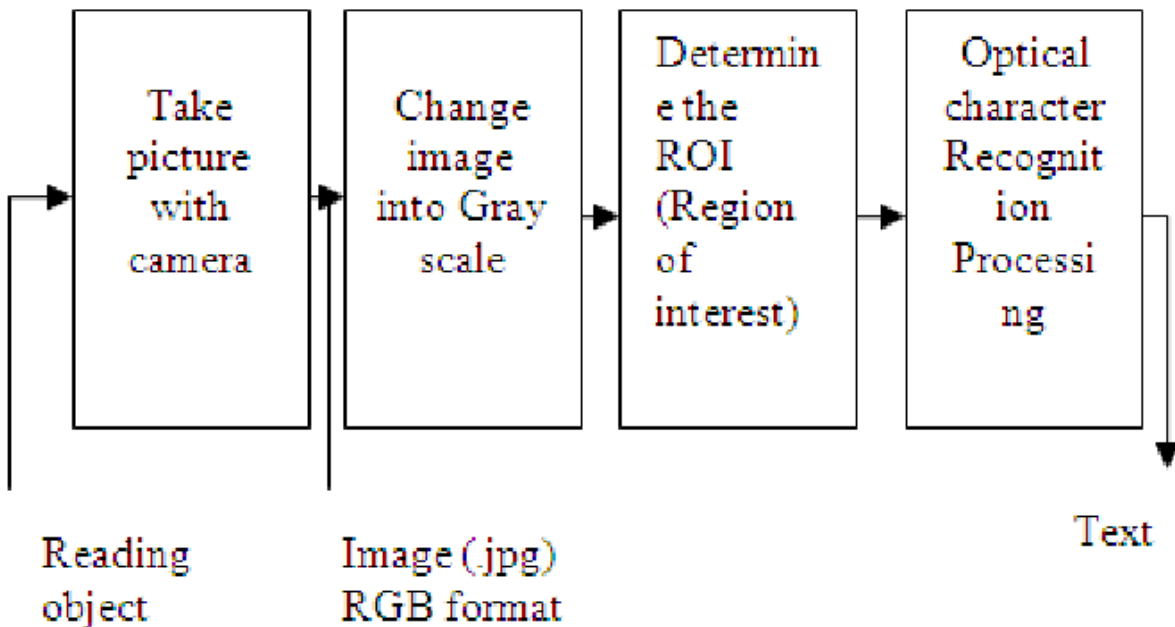
Figure 13 shows the flowchart of OCR system.



**Fig.13.** Flowchart of OCR system

## Software Design

Software processes the input image and converted into text format. The software implementation is showed in Figure 12.



**Fig.14.**Software design of image processing module

## The Voice Processing Module

In this module text is converted to speech. The output of OCR is the text, which is stored in a file (speech. txt). Here, Festival software is used to convert the text to speech. Festival is an open source Text To Speech (TTS) 7,8 system, which is available in many languages. In this project, English TTS 9–11system is used for reading the text.

**We Observed outcome:**

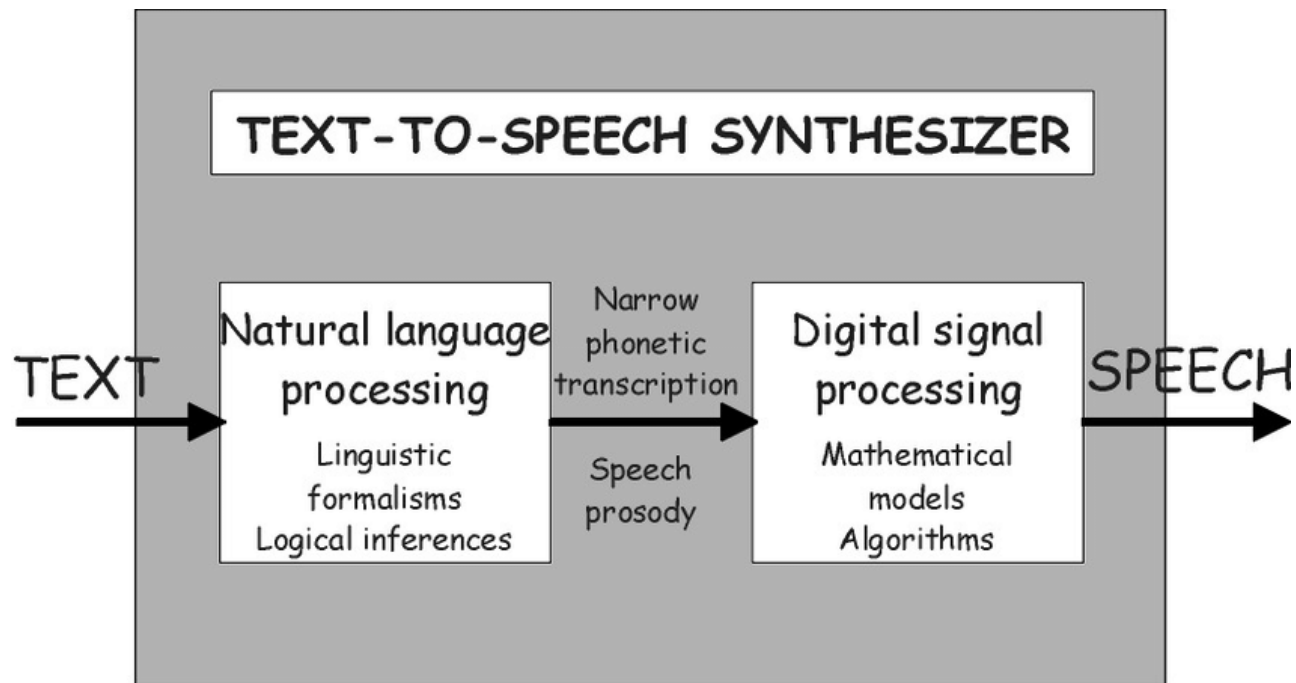
- Text is extracted from the image and converted to audio.
- It recognizes both capital as well as small letters.
- It recognizes numbers as well.
- Range of reading distance was 38-42cm.
- Character font size should be minimum 12pt.
- Maximum tilt of the text line is 4-5 degree from the vertical.

## **2.0.5 SPEECH SYNTHESIS**

Speech synthesis can be described as artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware. A text-to-speech (TTS) system converts normal language text into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output. The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written works on a home computer.

A text-to-speech system (or "engine") is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre processing, or tokenization. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences.

The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by



**Fig.15.** A simple but general functional diagram of a TTS system.

the front-end. The back-end—often referred to as the synthesizer—then converts the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.

There are different ways to perform speech synthesis. The choice depends on the task they are used for, but the most widely used method is Concatenative Synthesis, because it generally produces the most natural-sounding synthesized speech. Concatenative synthesis is based on the concatenation (or stringing together) of segments of recorded speech. There are three major sub-types of concatenative synthesis :

**Domain-specific Synthesis:** Domain-specific synthesis concatenates pre-recorded words and phrases to create complete utterances. It is used in applications where the variety of texts the system will output is limited to a particular domain, like transit schedule announcements or weather reports. The technology is very simple to implement, and has been in commercial use for a long time, in devices like talking clocks and calculators. The level of naturalness of these systems can be very high because the variety of sentence types is limited, and they closely match the prosody and intonation of the original recordings. Because these systems are limited by the words and phrases in their databases, they are not general-purpose and can only synthesize the combinations of words and phrases with which they have been pre-programmed. The blending of words within naturally spoken language however can still cause problems unless many variations are taken into account. For example, in nonrhotic dialects of English the "r" in words like "clear" /'klɪə/ is usually only pronounced when the following word has a vowel as its first letter (e.g. "clear out" is realized as /,klɪəɹ'ʌʊt/). Likewise in French, many final consonants become no longer silent if followed by a word that begins with a vowel, an effect called liaison. This alternation cannot be reproduced by a simple word-concatenation system, which would require additional complexity to be context-sensitive. This involves recording the voice of a person speaking the desired words and phrases. This is useful if only the restricted volume of phrases and sentences is used and the variety of texts the

system will output is limited to a particular domain e.g. a message in a train station, whether reports or checking a telephone subscriber's account balance. .

**Unit Selection Synthesis:** Unit selection synthesis uses large databases of recorded speech. During database creation, each recorded utterance is segmented into some or all of the following: individual phones, diphones, half-phones, syllables, morphemes, words, phrases, and sentences. Typically, the division into segments is done using a specially modified speech recognizer set to a "forced alignment" mode with some manual correction afterward, using visual representations such as the waveform and spectrogram. An index of the units in the speech database is then created based on the segmentation and acoustic parameters like the fundamental frequency (pitch), duration, position in the syllable, and neighboring phones. At runtime, the desired target utterance is created by determining the best chain of candidate units from the database (unit selection). This process is typically achieved using a specially weighted decision tree.

Unit selection provides the greatest naturalness, because it applies only a small amount of digital signals processing (DSP) to the recorded speech. DSP often makes recorded speech sound less natural, although some systems use a small amount of signal processing at the point of concatenation to smooth the waveform.

The output from the best unit-selection systems is often indistinguishable from real human voices, especially in contexts for which the TTS system has been tuned. However, maximum naturalness typically require unitselection speech databases to be very large, in some systems ranging into the gigabytes of recorded data, representing dozens of hours of speech. Also, unit selection algorithms have been known to select segments from a place that results in less than ideal synthesis (e.g. minor words become unclear) even when a better choice exists in the database.

**Diphone Synthesis:** Diphone synthesis uses a minimal speech database containing all the diphones (sound-to-sound transitions) occurring in a language. The number of diphones depends on the phonotactics of the language: for example, Spanish has about 800 diphones, and German about 2500. In diphone synthesis, only one example of each diphone is contained in the speech database. At runtime, the target prosody of a sentence is superimposed on these minimal units by means of digital signal processing techniques such as linear predictive coding, PSOLA or MBROLA. The quality of the resulting speech is generally worse than that of unit-selection systems, but more natural-sounding than the output of formant synthesizers. Diphone synthesis suffers from the sonic glitches of concatenative synthesis and the robotic-sounding nature of formant synthesis, and has few of the advantages of either approach other than small size. As such, its use in commercial applications is declining, although it continues to be used in research because there are a number of freely available software implementations.



## Structure of A Text-To-Speech Synthesizer System

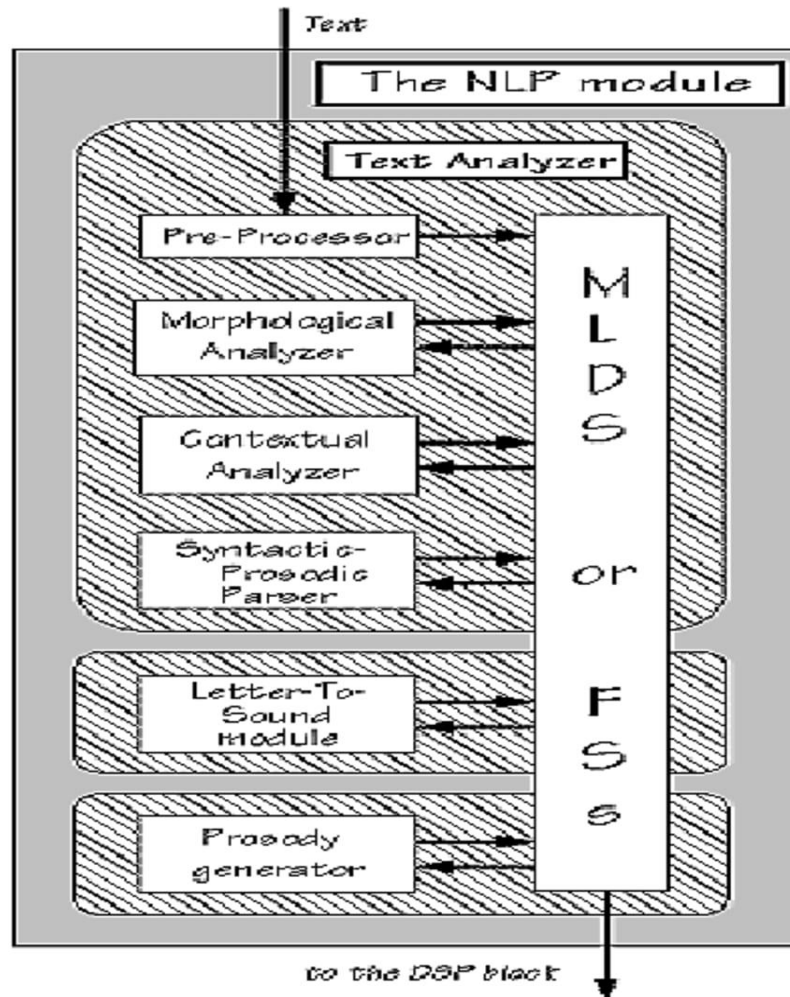
Text-to-speech synthesis takes place in several steps. The TTS systems get a text as input, which it first must analyze and then transform into a phonetic description.

Then in a further step it generates the prosody. From the information now available, it can produce a speech signal. The structure of the text-to-speech synthesizer can be broken down into major modules:

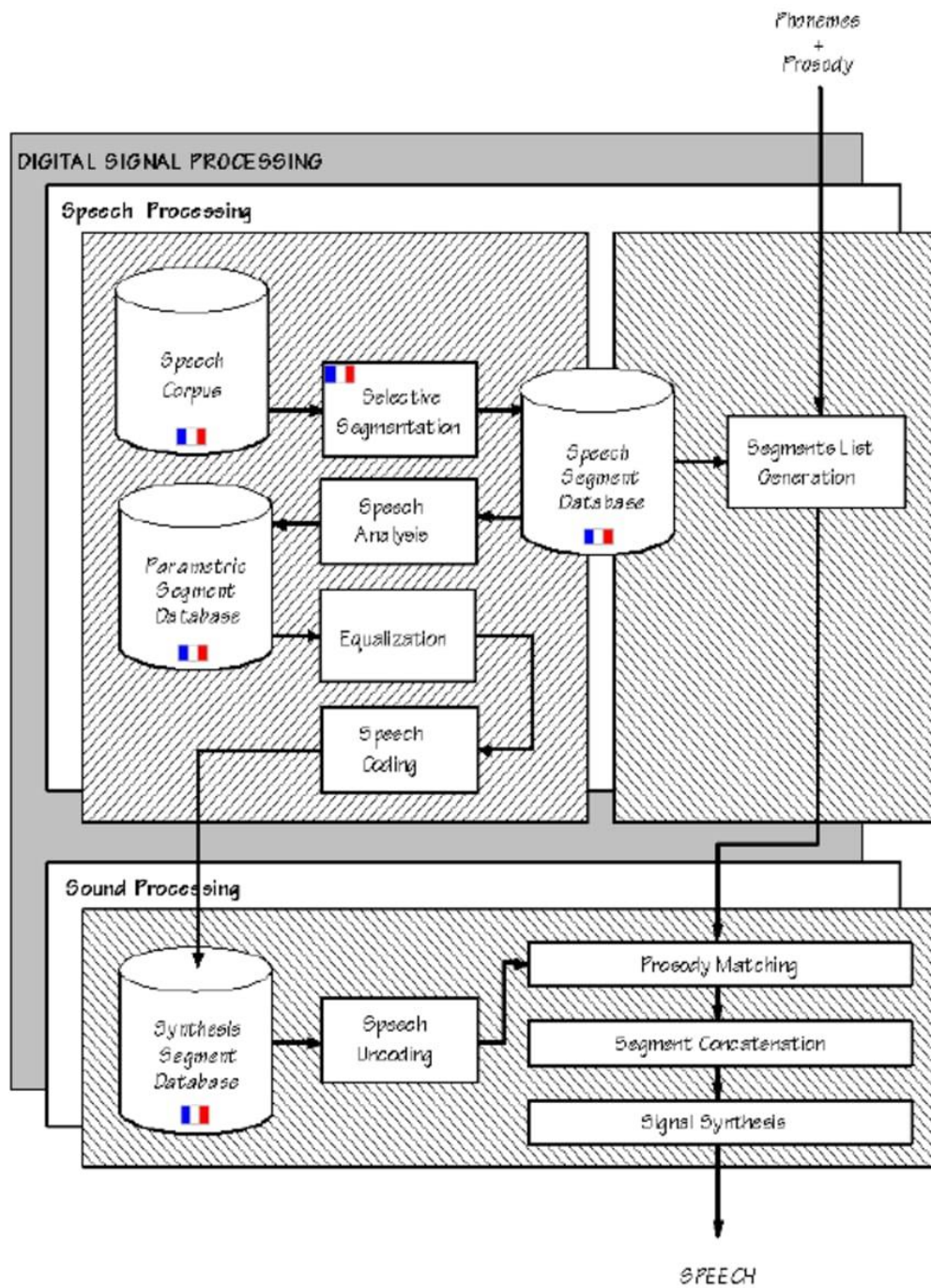
- **Natural Language Processing (NLP) module:** It produces a phonetic transcription of the text read, together with prosody.
- **Digital Signal Processing (DSP) module:** It transforms the symbolic information it receives from NLP into audible and intelligible speech. The major operations of the NLP module are as follows:
- **Text Analysis:** First the text is segmented into tokens. The token-to-word conversion creates the orthographic form of the token. For the token “Mr” the orthographic form “Mister” is formed by expansion, the token “12” gets the orthographic form “twelve” and “1997” is transformed to “nineteen ninety seven”.

- **Application of Pronunciation Rules:** After the text analysis has been completed, pronunciation rules can be applied. Letters cannot be transformed 1:1 into phonemes because correspondence is not always parallel. In certain environments, a single letter can correspond to either no phoneme (for example, “h” in “caught”) or several phoneme (“m” in “Maximum”). In addition, several letters can correspond to a single phoneme (“ch” in “rich”). There are two strategies to determine pronunciation:
  - In dictionary-based solution with morphological components, as many morphemes (words) as possible are stored in a dictionary. Full forms are generated by means of inflection, derivation and composition rules. Alternatively, a full form dictionary is used in which all possible word forms are stored. Pronunciation rules determine the pronunciation of words not found in the dictionary.
  - In a rule based solution, pronunciation rules are generated from the phonological knowledge of dictionaries. Only words whose pronunciation is a complete exception are included in the dictionary.The two applications differ significantly in the size of their dictionaries. The dictionary-based solution is many times larger than the rules-based solution’s dictionary of exception. However, dictionary-based solutions

can be more exact than rule-based solution if they have a large enough phonetic dictionary available.



**Fig.16.**Operations of the natural Language processing module of a TTS synthesizer.



**Fig.17.**The DSP component of a general concatenation based synthesizer

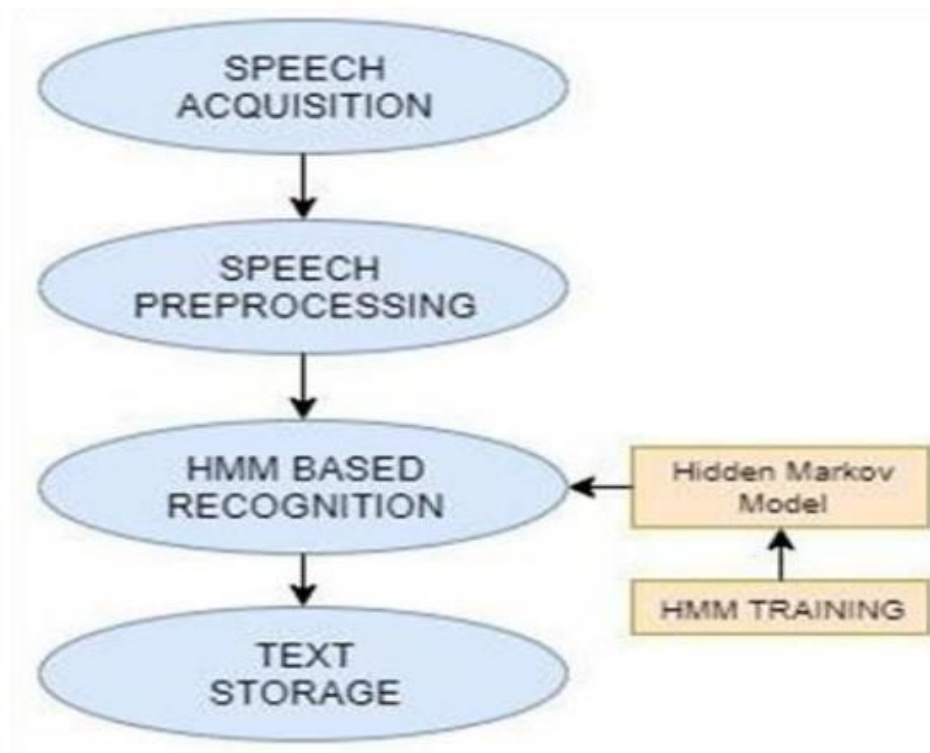
- **Prosody Generation:** after the pronunciation has been determined, the prosody is generated. The degree of naturalness of a TTS system is dependent on prosodic factors like intonation modelling (phrasing and accentuation), amplitude modelling and duration modelling (including the duration of sound and the duration of pauses, which determines the length of the syllable and the tempos of the speech).

The output (Figure 17) of the NLP module is passed to the DSP module. This is where the actual synthesis of the speech signal happens. In concatenative synthesis the selection and linking of speech segments take place. For individual sounds the best option (where several appropriate options are available) are selected from a database and concatenated.

## **2.0.6 MARKOV MODEL**

HMM are a sub class of the dynamic Bayesian models and exhibit their own kind of structure which makes them very useful for a large number of applications. The system that being modelled using the HMM has a Markov process having some hidden states and hence named as HMM. An HMM model in the most basic form can be assumed as a probabilistic model some state variable  $S$  and some observation variable  $O$  and transition between the states, each transition having an associated probability. In simple words, HMM belong to graphical model category that are efficient in predicting some hidden variables from some of visible variables. A basic example to quote on HMM is weather prediction for a location on the basis of the clothes that the people living there are wearing. The main advantage of using the HMM is Markov assumption that states “The future event is completely independent from the past event and depends only the present”. In simple words if we are aware of the current state, we need not have the training data to predict our future state. The HMM are mostly deployed in reinforcement learning projects like Speech Recognition, Text Recognition, Robot Localization, Biological sequence analysis, Handwriting Recognition, Pattern Recognition, etc. STT and TTS are particularly HMM based problem.

Figure 18 represents basic steps for STT conversion using HMM. It consists of 4 steps:



**Fig.18.**STT using HMM

1. **Speech Acquisition:** Getting the Speech data as input. The speech is taken as input using the microphone and is stored in the memory.
2. **Speech preprocessing:** The noisy environment and disturbances in speech signal are removed and it results in actual speech for conversion. By voice activity detection, these pauses are removed from the voice input.
3. **HMM based recognition:** The HMM is built by the system for each

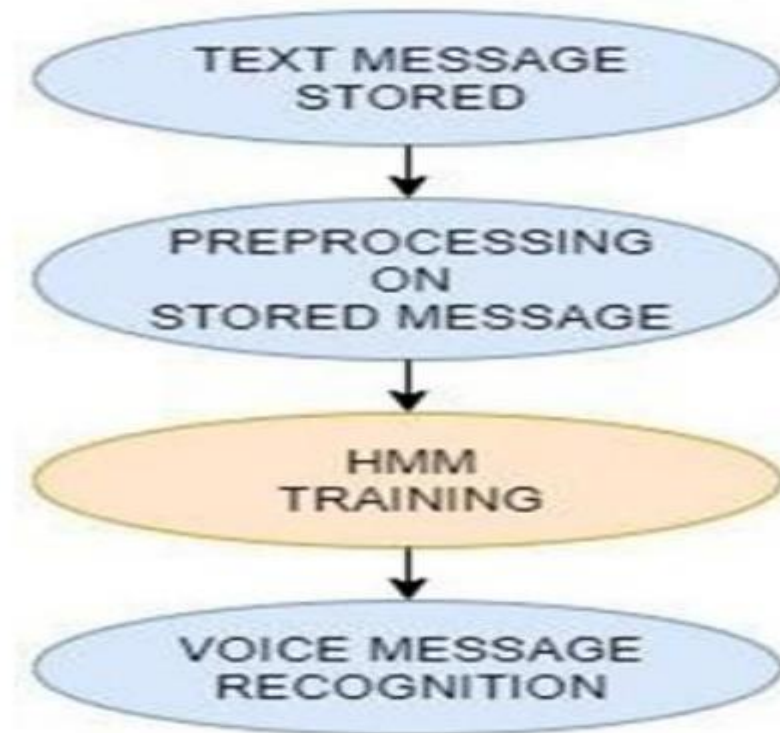
word in the vocabulary and these models are trained at the time of training phase. From speech preprocessing to HMM model construction, the steps of training are performed and generated HMM is loaded.

4. **Text Storage:** The matched text is stored as output and assembled to generate the text output. The model for TTS conversion is also implemented through HMM training for effective results

Figure 19 represents the basic steps for TTS conversion using HMM. It consists of 4 steps:

1. **Text Message Storage:** The text is taken as input at runtime from the user and stored into the system.
2. **Pre-processing of Stored message:** This step is useful for removing the irregularities and noisy data. Also, the text input splits into the different overlapping text frames.





**Fig.19.**TTS using HMM

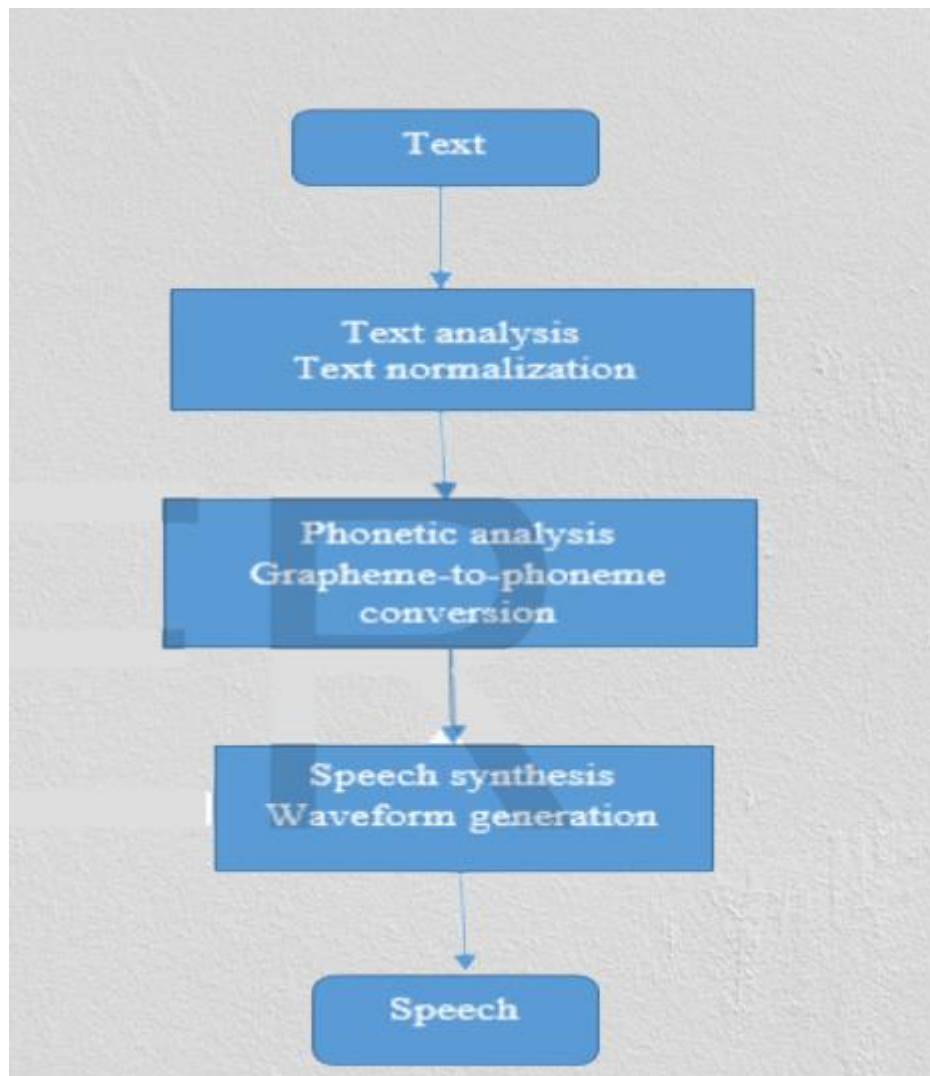
3. **HMM training:** HMM is used for text recognition and is a statistical model for modelling an unknown system using an observed output sequence. The HMM training involves pattern recognition for mapping the words to their sounds and creation of pattern representation of the features extracted from text class using one or more test patterns that correlates to speech sound of the same class. 4.Voice Message Recognition: Finally, the speech is generated as output and assembled which results in the voice output. The reason for

adapting HMM for both the TTS and STT processes is that it is a statistical model and provides a simple and effective framework for modeling temporal vector sequences. HMM provides a natural framework for building such models [14]. HMM depends on currently executing events and does not depends on the previous events. Also, the HMM is a really Dynamic Model which has training and self- adapting capabilities that increases the quality to STT and TTS. The HMM lies in the heart of all modern STT and TTS recognition systems.

## **2.0.7 Text-To-Speech synthesis:**

The main motto of TTS is to convert arbitrary text into waveform. Speech generation is generation of an acoustic wave form corresponding text and each of these units in the sequence. This involved text analysis, text normalization, text processing, grapheme-to phoneme conversion and speech synthesis [1]. Text analysis which analyse the input text such as dividing the text into words and sentences. Text normalization is the transformation of text into the pronounceable form, It is the front end of TTS that assign phonetic transcription to each and every word [3]-[4]. The process of assigning phonetic transcription to word is called grapheme to phoneme conversion. The phonetic analysis also known as word analysis focuses on phone within the word [5].finally symbolic linguistic representation produce sound.

Fig1 shows the flow diagram of TTS.



**Fig.20.** Block diagram of speech synthesis

### **Challenges of text to speech synthesis Speech synthesis**

Speech synthesis has developed for over the various systems, it has been integrated into several applications. Developing speech synthesis is a complicated process. There are many goals in TTS, the main goal of TTS is providing the high quality speech to users.

- 1) The development of computer based multi voice TTS system.
- 2) Developing of TTS system requires the knowledge about the language and produce human like speech.
- 3) The goals in building a computer system capable of speaking are to first build a system that clearly gets across the message.
- 4) The system is able to take any written input, if we build an English text-to-speech system, it should be capable of reading any English sentence given to it.
- 5) TTS system has made a good testing ground for many models and theories.

## **TTS SYSTEM MODEL**

For our understanding how the text to speech conversion by computer carried out, we define common form model and several other models.

The Common Form model: In the common form model, there are two components, a text analysis system which decodes the text signal and uncovers the form, and a speech synthesis system which encodes this form as speech. The first system is one of resolving ambiguity from a noisy signal so as to find a clean, unambiguous message; the second system is one where we take this message and

encode it as a different, noisy, ambiguous and redundant signal. In the basic common form model, we always read the words as they are encoded in the text; every word is read in the same order we encounter it. The key features of the model are

- The two fundamental processes text analysis and speech synthesis
- The task of analysis is find the form that is words from the text.
- The task of synthesis is to generate the signal from this form.
- Signal to signal model: In this model the process of converting written signal to spoken signal. Here we directly convert text to speech. In such models, the process is not seen as one of uncovering a linguistic message from a written signal, and then synthesising from this, but as a process where we try and directly convert the text into speech. In particular, the system is not divided into explicit analysis and synthesis stages
- Pipelined models: signal to signal model implemented as a pipeline model, the process is like the passing representation from one module to another. These type of systems are highly modular. Such that each module's job is defined as reading one type of information and producing another. Each module perform specific task such as speech tagging or pause insertion and so on. Modules are not explicitly linked that's why different theories and techniques are coexist.

- Grapheme and phoneme form model: This process is similar to common form model in that first a grapheme form of the text input is found, and it is converted to a phoneme form for synthesis. In this model grapheme form of the text input is -converted into phoneme form of speech synthesis that is exact pronunciation of each word of the input sentences. Words are not central to the representation as is the case in the common form model. This approach is particularly attractive in languages where the grapheme-phoneme correspondence is relatively direct; in such languages finding the graphemes often means the phonemes and hence pronunciation can accurately be found. For other languages such as English, this is more difficult, and for languages such as Chinese this approach is probably impossible. If it can be performed, a significant advantage of the grapheme/phoneme form model is that, an exhaustive knowledge of the words in a language is not necessary; little or no use is needed for a lexicon.

## **TEXT TO SPEECH SYNTHESIS METHODS**

The most differences usually are how the speech signal is generated from text. Among all the methods the easiest for understanding is concatenate speech synthesis. The audio signal is formed by concatenating pre-recorded speech samples. Another method that more closely related to articulatory speech synthesis is formant synthesis. There are several methods for synthesizing the speech. In this survey we explain three main speech synthesis method. Formant synthesis, articulatory speech synthesis, concatenative speech synthesis, HMM based speech synthesis

## **Formant Synthesis**

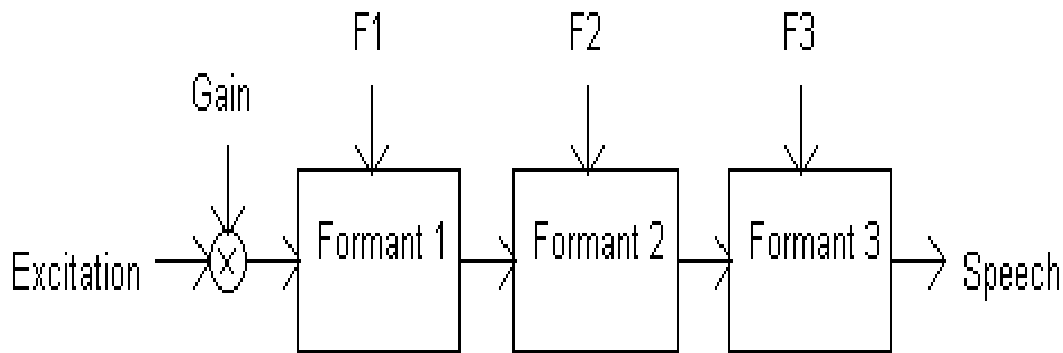
Formant synthesis was the first genuine synthesis technique and it was the dominant technique until the early 1980s. Formant synthesis also called by synthesis by rule it produce quality speech which sounds the unnatural. Formant synthesis adopt model based, acoustic phonetic approach to the synthesis problem. The filter in a formant synthesizer is typically implemented using cascade or parallel second-order filter sections, one per formant. Virtually in all formant synthesisers, the oral and nasal cavities are modelled as per parallel systems. Most modern rule-based text-to-speech systems descended from software based on this type of synthesis model.

In this approach, at least three formants are generally required to produce intelligible speech and to produce high quality speech up to five formants are used. The basic cascade format synthesis

- (i) A Cascade Formant Synthesizer
- (ii) A Parallel Formant Synthesizer
- (iii) Klatt Formant Synthesizer

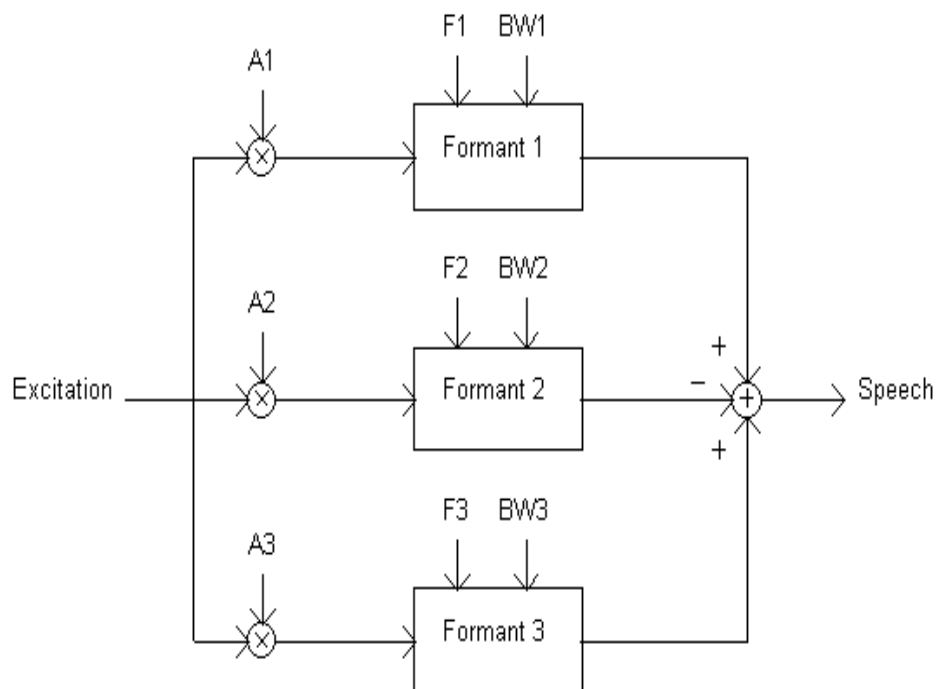


A cascade formant synthesizer Fig4.1 consist of band pass resonaters connected in series and the output of each formant synthesizer is applied to the input of the next one. A cascade formant synthesizer is good for non-nasal voice sounds rather than parallel formant synthesizer because it needs less control information.



**Fig.21.** Basic structure of cascade formant synthesizer

In a parallel cascade synthesizer consist of resonaters that are connected in parallel sometimes extra resonaters are used for nasal. The parallel structure enables controlling of bandwidth and gain for each formant individually, and also need more control information. The excitation signal is applied to all formants, and output are summed. Adjacent output of formant synthesizer must be summed in opposite phase to avoid unwanted zeros. Parallel structure is better for nasal and stop consonants. But some vowels can't be model with parallel formant synthesizer it is well with the cascade formant synthesizer.



**Fig.22.** Basic structure of a parallel formant synthesizer

klatt formant synthesizer introduced by Dennis klatt. The quality of klatt format synthesizer was very good for future views. The model has been incorporated into many TTS systems. Such as MLTALK, DECtalk, and Klattalk [24]. Klatt formant synthesis is a synthesis technique, where set of parameters generated from text from which the waveform is built from a cascade of modules to give the resultant signal.

### **Articulatory speech synthesis**

The most obvious way to synthesise speech is to try direct simulation of human speech production. This approach is called as an articulatory speech synthesis. The machine was mechanical device with tubes. The device is of course mimicking vocal

tract using sources and filters. Articulatory synthesis is the production of speech sounds using the model of vocal tract. Which directly simulates the movement of speech articulators. Two difficulties that arise in articulatory synthesis is how to generate the control parameters from the specification and how to find the right balance between highly accurate model that closely follows human physiology. It produce complete synthetic output based on mathematical models of the structure.

i) Module for generation of vocal tract movements (control model). (ii) A module for converting this movement information into a continuous succession of vocal tract geometries (vocal tract model), and (iii) a module for the generation of acoustic signals on the basis of this articulatory information (acoustic model)

(i) Vocal Tract Models:

(ii) Acoustic Models

(iii) Glottis Models

(iv) Noise Source Models

### **Concatenative Speech synthesis**

Concatenative synthesis produces artificial speech by concatenating the pre-recorded units of speech such as phonemes, diphones, syllables, words or sentences. The size of speech unit stored affects the quality of the synthesized speech if large sentences are stored the speech synthesized will sound natural. Concatenative speech synthesis generate high quality synthesized speech . Here we focus on two

aspect of Concatenative speech synthesis, a search unit and speech database. In concatenative speech synthesis it takes small units of speech that is acoustically parameterised data and concatenate sequences of these small units together to produce waveform. Major factor influencing the quality of synthesized speech such as fundamental frequency, amplitude, speaking rate and the availability of speech units having appropriate prosody in the database.concatenative speech synthesis can done by three different methods.

- (i) Diaphone based synthesis
- (ii) Domain based synthesis and
- (iii) Unit selection based synthesis

## 2.1 Literature Review Summary

Table 2.1: Literature review summary

Year and citation	Article Title	Purpose of study	Tools / Software Used	Comparison of technique	Source (Journal / Conference)	Findings	Data set (if used)	Evaluation parameters
2004	Recognition of Noisy Speech Using Dynamic Spectral Subband Centroids	References	Use of spectral subband Centroid.	—	NILL	Inter link between speech to text	NILL	It showed that the new dynamic SSC coefficients are more resilient to noise than the MFCC features.
2015	Voice Recognition	References	MFCC, HMM,	—	NILL	Recognition	NILL	Its improvement in voice to speech by using MFCC.
2014	Text to speech	References	FreeTTS, JAVA,	—	NILL	Interactive Voice Response (IVR) systems	NILL	Shows an improvement in text to speech.

2010	Speech Recognition Using HMM with MFCC-an analysis using Frequency Spectral Decomposition Technique	References	Resolution Decomposition with Separating Frequency is the mapping.	–	NILL	Recognition	NILL	It show an improvement in the quality metrics of speech recognition with respect to computational time, learning accuracy for a speech recognition system
2012	Speech Denoising using Different Types of Filters	Filters	FIR, IIR, WAVELETS, FILTER	–	NILL	Use of filters	NILL	Use of filter shows that estimation of clean speech and noise for speech enhancement in speech recognition.
2013	Isolated Speech Recognition using MFCC and DTW	MFC C	Dynamic Time Warping(DTW)	–	NILL	MFCC & DTW	NILL	It shows that the DTW is the best non linear feature matching technique in speech identification, with minimal error rates and fast computing speed.

2005	Speech Enhancement using Kalman filters for Restoration of shorttime DFT trajectories	References	Concept sequence modelling , two-level semantic-lexical modelling , and joint semantic-lexical modeling	–	NILL	Sematic lexical	NILL	Increase the semantic information utilized and tightness of integration between lexical and semantic items
2012	Speech Recognition with Hidden Markov Model	References	Hidden Markov Model	–	NILL	Markov	NILL	Develop a voice based user machine interface system
2012	Improving MFCCbased speech classification with FIR filter	References	FIR Filter	–	NILL	MFCC	NILL	Shows the improvement in recognition rates of spoken words

### **3 PROBLEM FORMULATION**

In our application we are combining Text-to-speech & Voice-to-text as single application, because now a days everyone using this Text-to-speech & Voice-to-text for their daily activities but they can't find both at a time .So , Our aim to build that every can use this application at a time with very easy steps. And here we are creating front page using HTML, CSS. Here we face a problem by API's because we are using different kind of API's like Web Speech API , SpeechSynthesis, speechRecognition .So we learn about different types of API's and completed the project successfully.



## 4 OBJECTIVES

In this project we have use Web Server API for the conversion of text-to-speech and from voice-to-speech. While using Web Server API it uses SpeechSynthesis (text-to-speech) and VoiceRecogniton (voice-to-text) using Polymer. The Web Speech API has a main controller interface for this — SpeechRecognition — plus a number of closely-related interfaces for representing grammar, results, etc. Generally, the default speech recognition system available on the device will be used for the speech recognition — most modern OSes have a speech recognition system for issuing voice commands.

We decided to do application based project that will help to convert Text-to-speech & Voice-to-text and we divided work into following objectives.

1. Creating frontend page.
2. Learn various tools to bulid Speech-To-Text.
3. Learn various tools to bulid Voice-To-Speech.
4. Tried many types of API's which gives accuracy to convert STT or VTS.

## 5 METHODOLOGY

In this project we have analyzed the existing system for speech recognition, speech to text conversion, speech to text conversion and SpeechSynthesis.

### Application Programming Interface

API(Application Programming Interface) is an interface that defines interactions between multiple Software applications as in figure 3. It defines the kinds of requests that can be made, how to make them, the data formats that should be used, conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionalities in various ways.

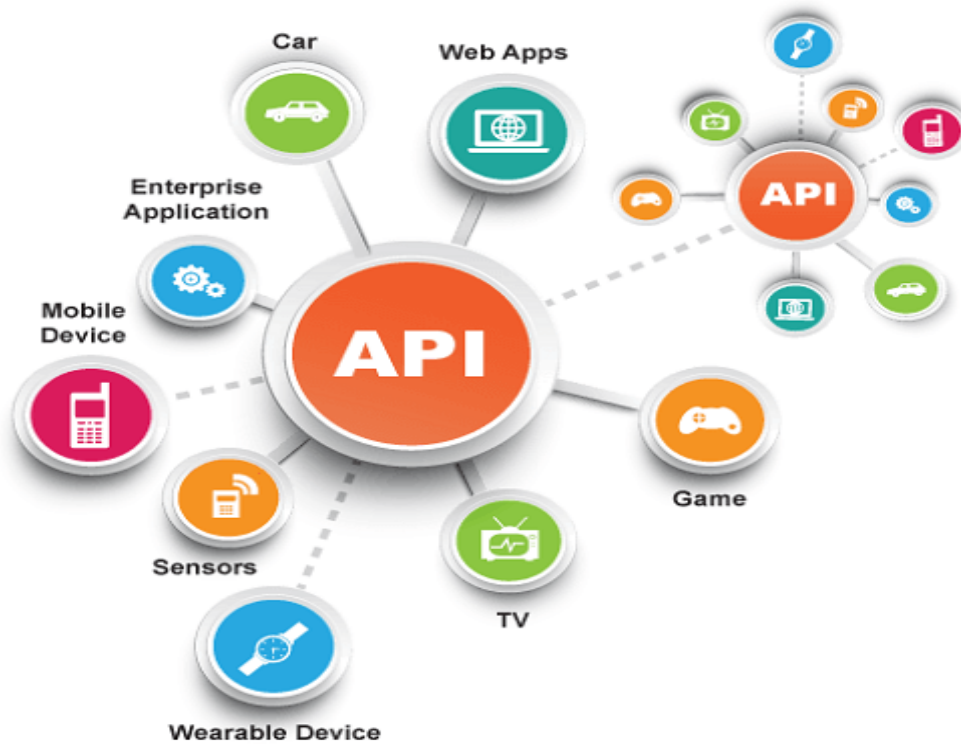
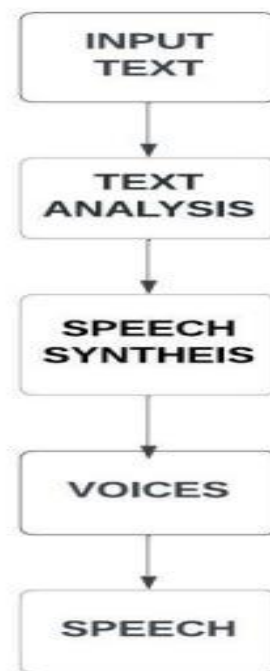


Fig.23. API uses

## Text-To-Speech Method

Our primary focus in this part is the Web Speech API which belongs to the Web/Browser API and Speech Synthesis.

The figure 4 shows all the steps involved in the text to speech conversion but the main phases of TTS systems are:



**Fig.24.** TTS System Flow

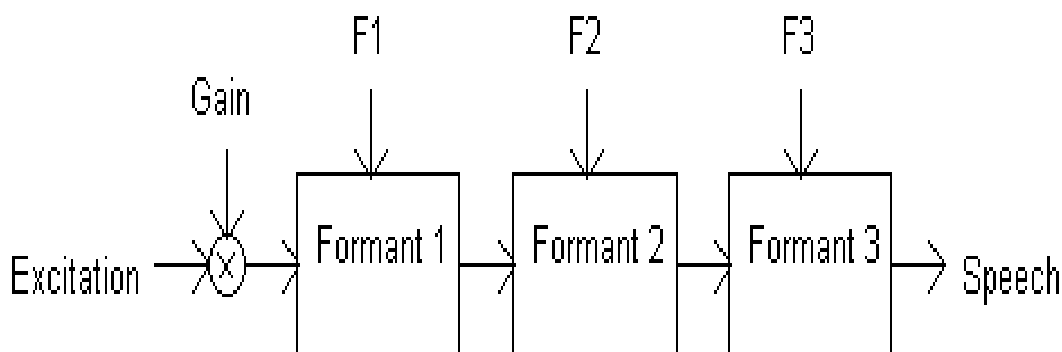
**Text Processing:** The input text is analysed, normalized (handles acronyms and abbreviation and match the text) and transcribed into phonetic or linguistic representation.

## SpeechSynthesis

The **Speech Synthesis** interface of the Web Speech API the controller interface for the speech service; this can be used to retrieve information about the synthesis voices available on the device, start and pause speech, and other commands besides. Some of the speech synthesis techniques are [5]:

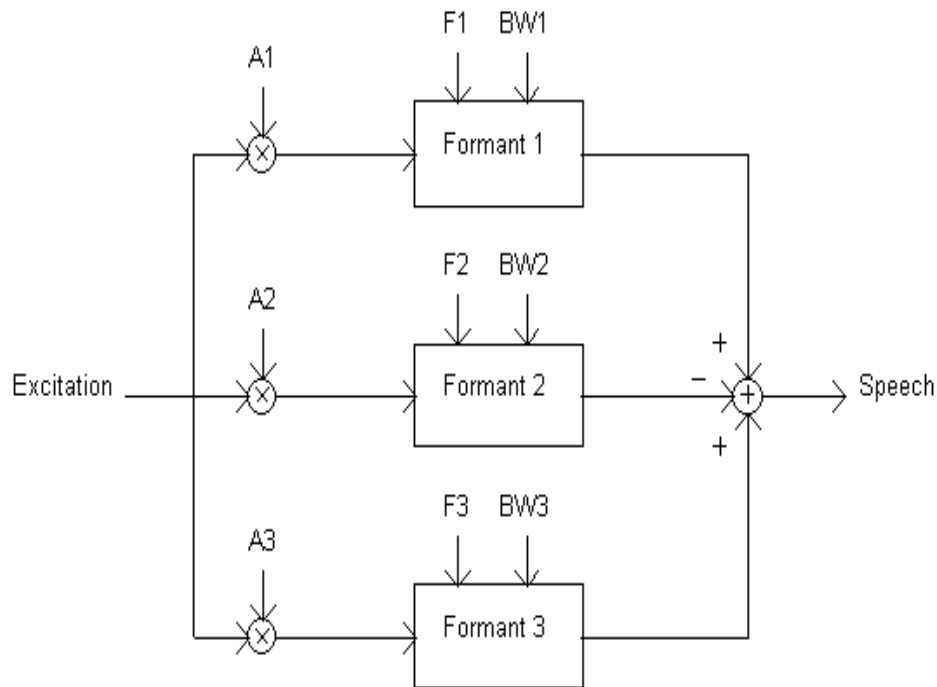
- i. **Articulator Synthesis:** Uses mechanical and acoustic model for speech generation. It produces intelligible synthetic speech but it is far from natural sound and hence not widely used.
- ii. **Formant Synthesis:** In this system, representation of individual speech segments are stored on a parametric basis. There are two basic structures in formant synthesis, parallel and cascade, but for better performance, some kind of combination of these 2 structures is used.

A cascade formant synthesizer consists of band-pass resonators connected in series. The output of each formant resonator is applied to the input of the successive one. The cascade structure needs only formant frequencies as control information.



**Fig.25.** Basic structure of cascade formant synthesizer

A parallel formant synthesizer consists of resonators connected in parallel. The excitation signal is applied to all formants simultaneously and their outputs are summed. [5]



**Fig.26.**Basic structure of a parallel formant synthesizer

- iii. **Concatenative Synthesis:** This technique synthesizes sound by concatenating short samples of sound called units. It is used in speech synthesis to generate user specific sequence of sound from a database built from the recording of other sequences. Units for Concatenative synthesis are [5]: Phone- a single unit of sound; Diphone is defined as the signal from either midpoint of a phone or point of least change within the phone to the similar point in the next phone; Triphone- is a section of the signal taking in a sequence going from middle of a phone completely through the next one to the middle of a third.

## Voice function:

The voices array contains all of the available voices in the Browser Web speech API, and the `getVoices` function returns all of the available voices, along with their names and languages. Finally, the voices are added to our website's drop-down menu of select options. The user will be able to choose their preferred voices.

```
13 // Init voices array
14 let voices = [];
15
16 ▼ const getVoices = () => {
17     voices = synth.getVoices();
18
19     // Loop through voices and create an option for each one
20 ▼ voices.forEach(voice => {
21     // Create option element
22     const option = document.createElement('option');
23     // Fill option with voice and language
24     option.textContent = voice.name + '(' + voice.lang + ')';
25
26     // Set needed option attributes
27     option.setAttribute('data-lang', voice.lang);
28     option.setAttribute('data-name', voice.name);
29     voiceSelect.appendChild(option);
30 });
31 };
32 |
33 getVoices();
34 ▼ if (synth.onvoiceschanged !== undefined) {
35     synth.onvoiceschanged = getVoices;
36 }
```

Fig.27.shows the voice working

## Speak Function:

We'll create some function talk. Talk handles speaking events by creating a wave background when a voice speaks, managing the selected voice, controlling the rate and pitch of the voice, and handling errors.

```
38 // Speak
39 ▾ const speak = () => {
40   // Check if speaking
41   ▾ if (synth.speaking) {
42     return;
43   }
44   ▾ if (textInput.value !== '') {
45     // Add background animation
46     body.style.background =
47       "#0acffe url(https://res.cloudinary.com/nonsoblip/image/upload/v1620236458/wave_n3rtre.gif)";
48     body.style.backgroundRepeat = 'repeat-x';
49     body.style.backgroundSize = '100% 100%';
50
51     // Get speak text
52     const speakText = new SpeechSynthesisUtterance(textInput.value);
53
54     // Speak end
55     ▾ speakText.onend = e => {
56       body.style.backgroundImage = `linear-gradient(to right, #0acffe 0%, #495aff 100%)`;
57       // body.style.background = '#141414';
58     };
59
60     // Speak error
61     ▾ speakText.onerror = e => {
62       console.error('Something went wrong');
63     };
64
65     // Selected voice
66     const selectedVoice = voiceSelect.selectedOptions[0].getAttribute(
67       'data-name'
68     );
69
70     // Loop through voices
71     ▾ voices.forEach(voice => {
72       ▾ if (voice.name === selectedVoice) {
73         speakText.voice = voice;
74       }
75     });
76     // Set pitch and rate
77     speakText.rate = rate.value;
78     speakText.pitch = pitch.value;
79     // Speak
80     synth.speak(speakText);
```

Fig.28.shows the speak-set rate&pitch

## **Audio Captcha**

This is a speech to text API that is specifically designed to understand audio generated by "Audio Captcha" mechanisms as in figure 2. It can take audio data as input.

## **Alerting Hub Batch send**

This API allows you to send multiple, different, alerts at the same time. In effect this API removes the need for you to call sendalert multiple times and is most often of use when specifying a set.

## **Alerting Hub Store Content**

The store content API allows you to upload either alerts, recipient lists or content data and store directly onto the system. Multiple content blocks can be included within a single transaction.

## **Word Text To Speech**

Free online Text-to-Speech (TTS) API supports 8 languages with 38 voices. Developers can get advantage of Woord's free text-to-speech online service for any platforms.

## **Coding Part :-**

Here we are going to attach all the coding part of Text – TO – Speech part .

Figure 13 & 14 shows the final output for Text To Speech.



```

text-to-speech index.html (proj main) - Brackets
File Edit Find View Navigate Debug Help

Working Files
main.html
maincss.css
main.js
app.js
style.css
ttttt.html
text-to-speech index.html
voice-to-text index.html

proj main
New folder
app.js
hloo.jpg
main.html
main.js
maincss.css
speech-recognition.png
Text to speech-01.jpeg
text-to-speech index.html
voice-to-text index.html

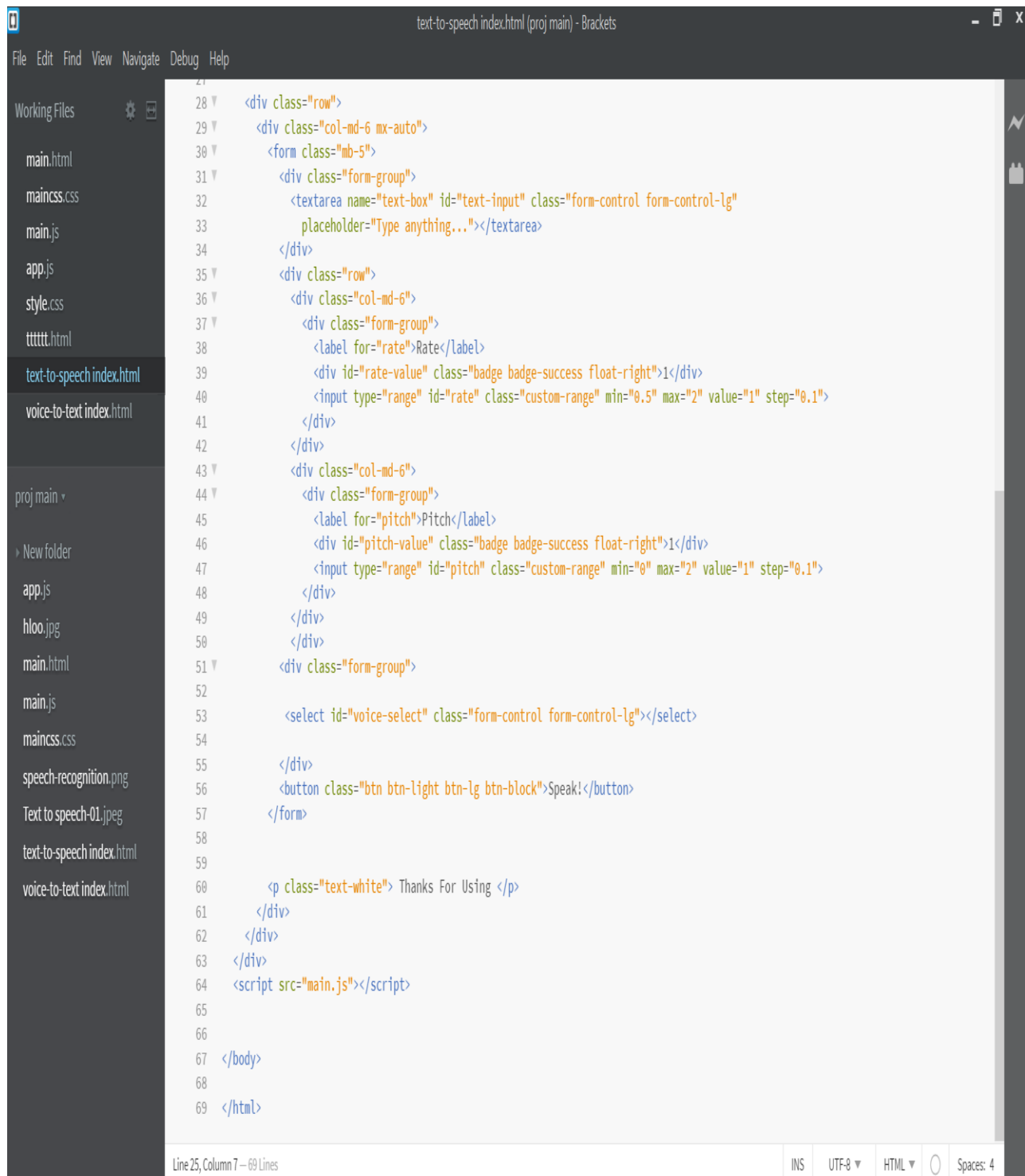
1 <html lang="en" class="js-focus-visible" data-js-focus-visible="">
2
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
8     integrity="sha384-MCW98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
9   <link rel="stylesheet" href="/css/style.css">
10  <title>Type-N-Speak</title>
11 </style>
12 <body {
13
14 }
15 </style>
16 </head>
17
18 <body class="text-white">
19   <div class="container text-center">
20     <div class="logo col-lg-5 col-md-6 col-sm-10 mx-auto my-4">
21
22       
24
25     </div>
26
27   <div class="row">
28     <div class="col-md-6 mx-auto">
29       <form class="mb-5">
30         <div class="form-group">
31           <textarea name="text-box" id="text-input" class="form-control form-control-lg"
32             placeholder="Type anything..."></textarea>
33         </div>
34       </div>
35     <div class="row">
36       <div class="col-md-6">
37         <div class="form-group">
38           <label for="rate">Rate</label>
39           <div id="rate-value" class="badge badge-success float-right">1</div>
40           <input type="range" id="rate" class="custom-range" min="0.5" max="2" value="1" step="0.1">
41         </div>
42       </div>

```

Line 25, Column 7 — 69 Lines

INS UTF-8 HTML Spaces: 4

**Fig.29.coding part of html-1**



The image shows a code editor window titled "text-to-speech index.html (proj main) - Brackets". The editor displays HTML code for a web page. The left sidebar shows a file explorer with the following files: main.html, maincss.css, main.js, app.js, style.css, tttttt.html, text-to-speech index.html (selected), voice-to-text index.html, proj main, New folder, app.js, hloo.jpeg, main.html, main.js, maincss.css, speech-recognition.png, Text to speech-01.jpeg, text-to-speech index.html, and voice-to-text index.html. The main editor area shows the following HTML code:

```
28 <div class="row">
29   <div class="col-md-6 mx-auto">
30     <form class="mb-5">
31       <div class="form-group">
32         <textarea name="text-box" id="text-input" class="form-control form-control-lg"
33           placeholder="Type anything..."></textarea>
34       </div>
35     <div class="row">
36       <div class="col-md-6">
37         <div class="form-group">
38           <label for="rate">Rate</label>
39           <div id="rate-value" class="badge badge-success float-right">1</div>
40           <input type="range" id="rate" class="custom-range" min="0.5" max="2" value="1" step="0.1">
41         </div>
42       </div>
43     <div class="col-md-6">
44       <div class="form-group">
45         <label for="pitch">Pitch</label>
46         <div id="pitch-value" class="badge badge-success float-right">1</div>
47         <input type="range" id="pitch" class="custom-range" min="0" max="2" value="1" step="0.1">
48       </div>
49     </div>
50   </div>
51   <div class="form-group">
52     <select id="voice-select" class="form-control form-control-lg"></select>
53   </div>
54   <button class="btn btn-light btn-lg btn-block">Speak!</button>
55 </form>
56
57 <p class="text-white"> Thanks For Using </p>
58 </div>
59 </div>
60 <script src="main.js"></script>
61
62 </body>
63 </html>
```

The status bar at the bottom indicates "Line 25, Column 7 — 69 Lines", "INS", "UTF-8", "HTML", and "Spaces: 4".

**Fig.30.coding part of html-2**

```
1  const synth = window.speechSynthesis;
2
3  // DOM Elements
4  const textForm = document.querySelector('form');
5  const textInput = document.querySelector('#text-input');
6  const voiceSelect = document.querySelector('#voice-select');
7  const rate = document.querySelector('#rate');
8  const rateValue = document.querySelector('#rate-value');
9  const pitch = document.querySelector('#pitch');
10 const pitchValue = document.querySelector('#pitch-value');
11 const body = document.querySelector('body');
12
13 // Init voices array
14 let voices = [];
15
16 const getVoices = () => {
17   voices = synth.getVoices();
18 }
19 // Loop through voices and create an option for each one
20 voices.forEach(voice => {
21   // Create option element
22   const option = document.createElement('option');
23   // Fill option with voice and language
24   option.textContent = voice.name + '(' + voice.lang + ')';
25
26   // Set needed option attributes
27   option.setAttribute('data-lang', voice.lang);
28   option.setAttribute('data-name', voice.name);
29   voiceSelect.appendChild(option);
30 });
31 };
32
33 getVoices();
34 if (synth.onvoiceschanged !== undefined) {
35   synth.onvoiceschanged = getVoices;
36 }
37
38 // Speak
39 const speak = () => {
40   // Check if speaking
41   if (synth.speaking) {
42     return;
```

Line 1, Column 1 — 101 Lines

INS UTF-8 JavaScript Spaces: 4

**Fig.31.**coding part of js-1

```
38 // Speak
39 const speak = () => {
40   // Check if speaking
41   if (synth.speaking) {
42     return;
43   }
44   if (textInput.value !== '') {
45     // Add background animation
46     body.style.background =
47       "#0acffe url(https://res.cloudinary.com/nonsoblip/image/upload/v1628236458/wave_n3rtre.gif)";
48     body.style.backgroundRepeat = 'repeat-x';
49     body.style.backgroundSize = '100% 100%';
50
51     // Get speak text
52     const speakText = new SpeechSynthesisUtterance(textInput.value);
53
54     // Speak end
55     speakText.onend = e => {
56       body.style.backgroundImage = 'linear-gradient(to right, #0acffe 0%, #495aff 100%)';
57       // body.style.background = '#141414';
58     };
59
60     // Speak error
61     speakText.onerror = e => {
62       console.error('Something went wrong');
63     };
64
65     // Selected voice
66     const selectedVoice = voiceSelect.selectedOptions[0].getAttribute(
67       'data-name'
68     );
69
70     // Loop through voices
71     voices.forEach(voice => {
72       if (voice.name === selectedVoice) {
73         speakText.voice = voice;
74       }
75     });
76
77     // Set pitch and rate
78     speakText.rate = rate.value;
79     speakText.pitch = pitch.value;
80     // Speak
```

Line 1, Column 1 - 101 Lines

INS UTF-8 JavaScript Spaces: 4

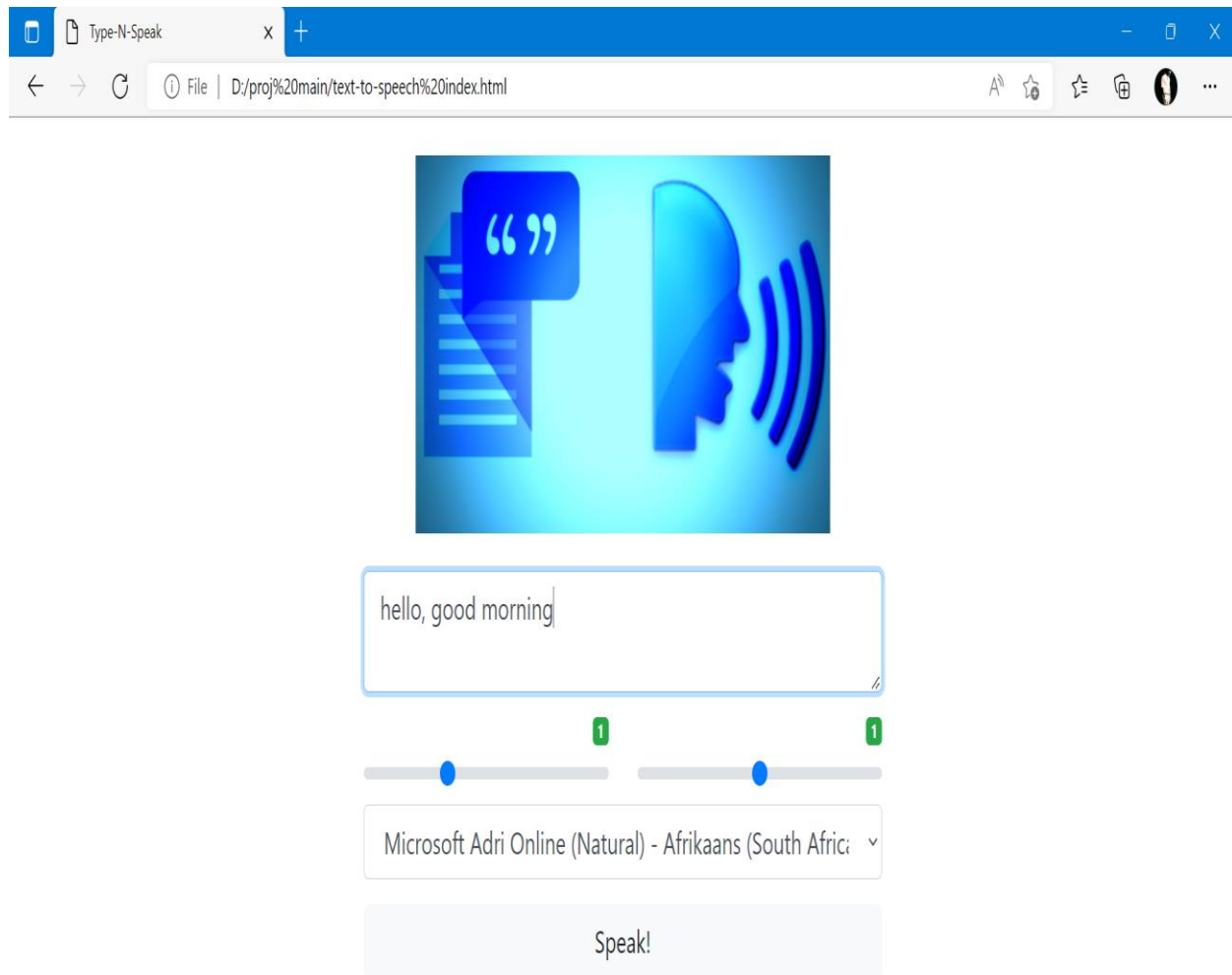
**Fig.32.**coding part of js-2

```
59
60 // Speak error
61 speakText.onerror = e => {
62     console.error('Something went wrong!');
63 };
64
65 // Selected voice
66 const selectedVoice = voiceSelect.selectedOptions[0].getAttribute(
67     'data-name'
68 );
69
70 // Loop through voices
71 voices.forEach(voice => {
72     if (voice.name === selectedVoice) {
73         speakText.voice = voice;
74     }
75 });
76
77 // Set pitch and rate
78 speakText.rate = rate.value;
79 speakText.pitch = pitch.value;
80 // Speak
81 synth.speak(speakText);
82 }
83 };
84
85 // EVENT LISTENERS
86
87 // Text form submit
88 textForm.addEventListener('submit', e => {
89     e.preventDefault();
90     speak();
91     textInput.blur();
92 });
93
94 // Rate value change
95 rate.addEventListener('change', e => (rateValue.textContent = rate.value));
96
97 // Pitch value change
98 pitch.addEventListener('change', e => (pitchValue.textContent = pitch.value));
99
100 // Voice select change
101 voiceSelect.addEventListener('change', e => speak());
```

Line 1, Column 1 — 101 Lines    INS    UTF-8    JavaScript    ⚠    Spaces: 4

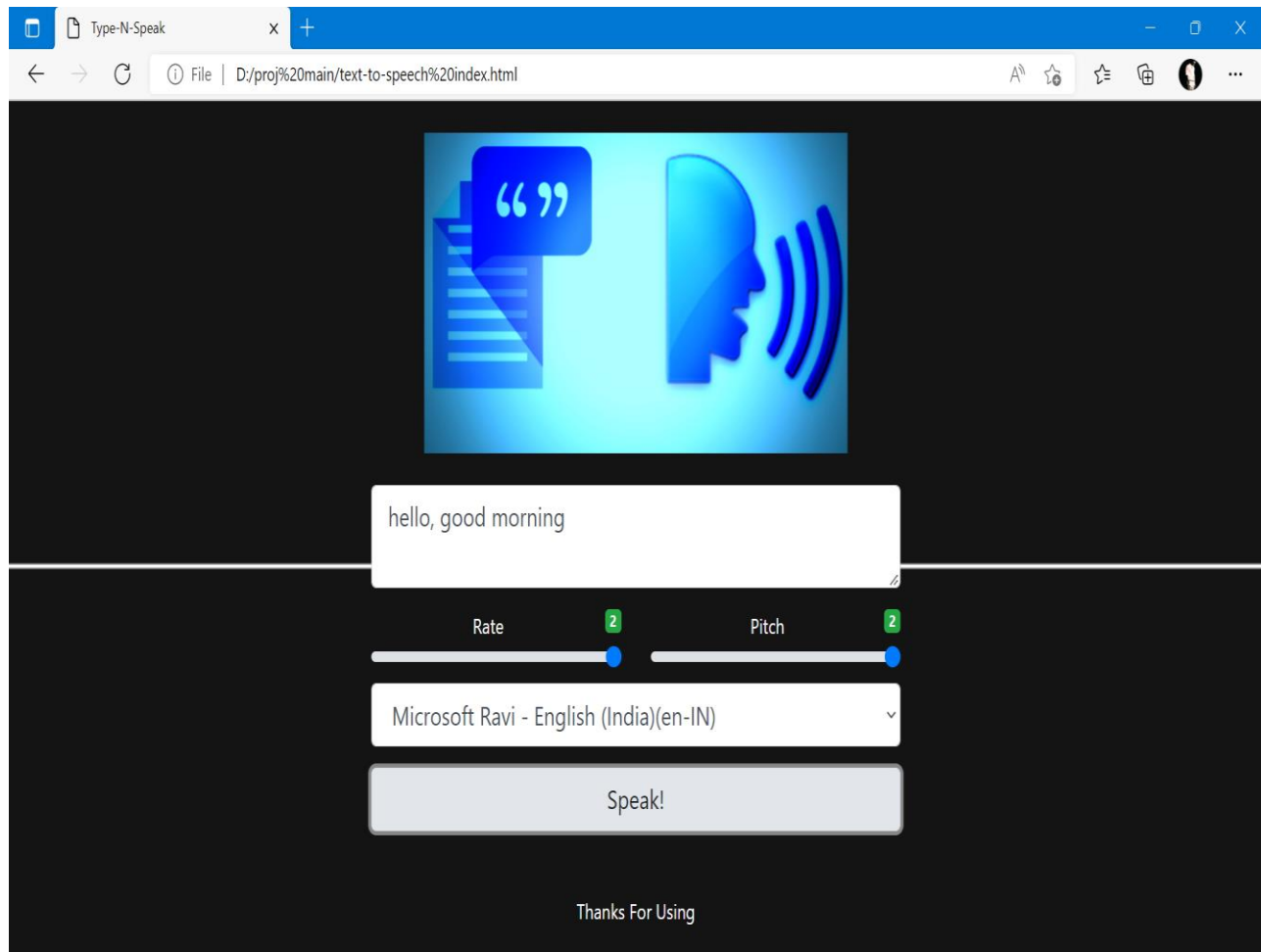
**Fig.33.**coding part of js-3

- In Text To Speech we allocate the text box which users can type or copy paste their text as shown in figure 13 and given options that user can set the pitch and rate as there interest. Now there are several languages as per user preference and click on Speak.



**Fig.34.**output of TTS

- This figure shows that when user enter text and select the rate & pitch.



**Fig.35.** final output of TTS

## Voice-To-Text

SpeechRecognition Web API, another component of the Web Speech API that is used to convert Voice to Texts, and then build a simple Web application that converts our words/sentences to text format like shown in figure 5.



Fig.36. voice to text

### SPEECH RECOGNITION:

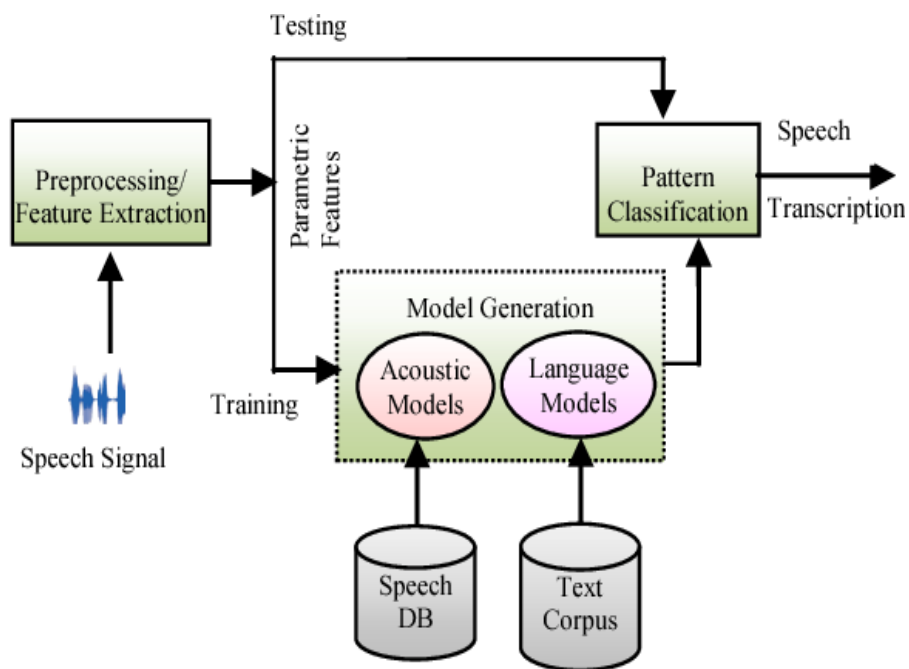
Speech Recognition is the ability of machine/program to identify words and phrases in spoken language and convert them into machine-readable format. Speech Recognition Systems can be classified here:

- **Speaker:** All speakers have a different kind of voice. The models hence are either designed for a specific speaker or an independent speaker.
- **Vocal Sound:** The way the speaker speaks also plays a role in speech recognition. Some models can recognize either single utterances or separate utterance with a pause in between.



- **Vocabulary:** The size of the vocabulary plays an important role in determining the complexity, performance, and precision of the system

Basic Speech Recognition Model: Each speech recognition system follow some standard steps as shown in figure 1.



**Fig.37.SpeechRecognitionsteps**

**Pre-processing:** The analog speech signal is transformed into digital signals for later processing. This digital signal is moved to the first order filters to spectrally flatten the signals. This helps in increasing the signal's energy

**Feature Extraction:** This step finds the set of parameters of utterances that have a correlation with speech signals. These parameters, known as features, are computed through processing of the acoustic waveform. The main focus is to compute a sequence of feature vectors (relevant information) providing a compact representation of the given input signal.

## **Accurate transcription**

The most important thing, regardless of what you're using STT for, is accurate transcription like figure 4. If you're getting back transcripts that look like MadLibs, it's unlikely you're going to get much business value from them. The absolute baseline accuracy for readable transcriptions is 80%.

## **Real-time streaming**

Again, not everyone will need real-time streaming. However, if you want to use VTT to create, for example, truly conversational AI that can respond to customer inquiries in real time, you'll need to use a VTT API that returns its results as quickly as possible.

## **Multi-language support**

If you're planning to handle multiple languages or dialects, this should be a key concern. And even if you aren't planning on multilingual support now, if there's any chance that you would in the future, you're best off starting with a service that offers many languages and is always expanding to more.

## **Automatic punctuation & capitalization**

Depending on what you're planning to do with your transcripts, you might not care if they're formatted nicely. But if you're planning on surfacing them publicly, having this included in what the STT API provides can save you time.

```

5
6 var SpeechRecognition = SpeechRecognition || webkitSpeechRecognition;
7 var SpeechGrammarList = SpeechGrammarList || webkitSpeechGrammarList;
8 var grammar = "#JSGF V1.0;";
9 var recognition = new SpeechRecognition();
10 var speechGrammarList = new SpeechGrammarList();
11
12 speechGrammarList.addFromString(grammar, 1);
13 recognition.grammars = speechGrammarList;
14 recognition.lang = "en-US";
15 recognition.interimResults = true;
16 recognition.continuous = true;
17
18
19 ▼ recognition.onresult = function (event) {
20     var transcript = Array.from(event.results)
21         .map((result) => result[0])
22         .map((result) => result.transcript)
23         .join("");
24     message.textContent = transcript;
25 };
26 ▼ recognition.onerror = function (event) {
27     message.textContent = "Error occurred in recognition: " + event.error;
28     recognizing = false;
29 };
30

```

**Fig.38.**Speechrecognition, voice, grammarobject

## Custom vocabulary

Being able to define custom vocabulary is helpful if your audio has lots of custom terms, abbreviations, and acronyms as shown in figure 17 that an off-the-shelf model wouldn't have been exposed to.

### Coding Part:-

Here we are going to attach all the coding part of Voice – TO – Text part .

Figure 13 & 14 shows the final output for Text To Speech.

```

1
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
8     integrity="sha384-MCW98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
9   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.2/css/all.min.css" />
10  <title>Speech-To-Text</title>
11  <style>
12    body {
13
14    }
15    #note {
16      font-size: 15px;
17      font-family: italic;
18    }
19    .icons{
20      font-size: 20px;
21    }
22    .pointer{
23      cursor: pointer;
24    }
25  </style>
26 </head>
27
28 <body class="text-white">
29   <div class="container text-center">
30     <div class="logo col-lg-5 col-md-6 col-sm-10 mx-auto my-5">
31
32       
34
35     </div>
36     <div class="row">
37       <div class="col-md-6 mx-auto">
38         <form class="mb-5">
39           <div class="form-group">
40             <textarea name="text-box" id="note" rows="8" value="" class="form-control form-control-lg"
41               placeholder="Type anything..." disabled></textarea>
42           </div>

```

Line 33, Column 35 - 63 Lines

INS UTF-8 HTML Spaces: 4

**Fig.39.**coding part of html-1

```

voice-to-text index.html (proj main) - Brackets
File Edit Find View Navigate Debug Help
Working Files
main.html
maincss.css
main.js
app.js
style.css
ttttt.html
text-to-speech index.html
voice-to-text index.html
proj main
New folder
app.js
hloo.jpg
main.html
main.js
maincss.css
speech-recognition.png
Text to speech-01.jpeg
text-to-speech index.html
voice-to-text index.html
21
22 .pointer{
23     cursor: pointer;
24 }
25 </style>
26 </head>
27
28 <body class="text-white">
29     <div class="container text-center">
30         <div class="logo col-lg-5 col-md-6 col-sm-10 mx-auto my-5">
31
32             
34
35         </div>
36         <div class="row">
37             <div class="col-md-6 mx-auto">
38                 <form class="mb-5">
39                     <div class="form-group">
40                         <textarea name="text-box" id="note" rows="8" value="" class="form-control form-control-lg"
41                             placeholder="Type anything..." disabled></textarea>
42                     </div>
43                     <div class="icon-wrap d-flex justify-content-around">
44                         <span class="text-dark" id="copyTxt">
45                             <i class="fas fa-copy text-primary icons pointer"></i></span>
46                         <span id="clearNote"><i class="fas fa-trash-alt text-danger icons pointer"></i></span>
47                     </div>
48                     <div class="btn-wrap d-flex justify-content-around mt-5">
49                         <span class="pointer" id="micStart">
50                             
53                             </span>
54                         </div>
55                     </form>
56                     <p class="text-dark"> Thanks for using
57                     </p>
58                 </div>
59             </div>
60         </div>
61         <script src="./app.js"></script>
62     </body>
63 </html>
Line 33, Column 35 – 63 Lines
INS UTF-8 HTML Spaces: 4

```

**Fig.40.coding part of html-2**

```
1 var message = document.querySelector("#note");
2 var micStart = document.querySelector("#micStart");
3 var copy__Text = document.querySelector("#copyText");
4 var clear__Note = document.querySelector("#clearNote");
5
6 var SpeechRecognition = SpeechRecognition || webkitSpeechRecognition;
7 var SpeechGrammarList = SpeechGrammarList || webkitSpeechGrammarList;
8 var grammar = "#JSGF V1.0;";
9 var recognition = new SpeechRecognition();
10 var speechGrammarList = new SpeechGrammarList();
11
12 speechGrammarList.addFromString(grammar, 1);
13 recognition.grammars = speechGrammarList;
14 recognition.lang = "en-US";
15 recognition.interimResults = true;
16 recognition.continuous = true;
17
18
19 recognition.onresult = function (event) {
20     var transcript = Array.from(event.results)
21         .map((result) => result[0])
22         .map((result) => result.transcript)
23         .join("");
24     message.textContent = transcript;
25 };
26 recognition.onerror = function (event) {
27     message.textContent = "Error occurred in recognition: " + event.error;
28     recognizing = false;
29 };
30
31 const copyPaseText = () => {
32     var copyText = document.querySelector("#note").value;
33     navigator.clipboard.writeText(copyText).then(
34         function () {
35             alert(" Copying to clipboard was successful!");
36         },
37         function (err) {
38             console.error("Async: Could not copy text: ", err);
39         }
40     );
41 };
42 const clearNote = () => {
```

Line 1, Column 1 - 54 Lines

INS UTF-8 JavaScript Spaces: 4

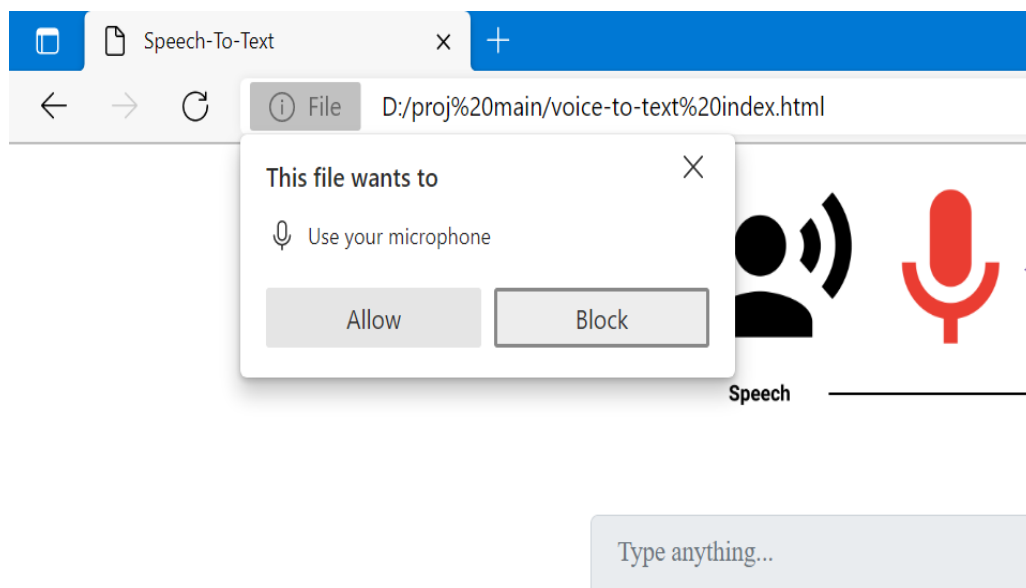
**Fig.41.**coding part of js-1

```
12 speechRecognitionList.addFunction(grammar, 1);
13 recognition.grammars = speechGrammarList;
14 recognition.lang = "en-US";
15 recognition.interimResults = true;
16 recognition.continuous = true;
17
18
19 recognition.onresult = function (event) {
20     var transcript = Array.from(event.results)
21     .map((result) => result[0])
22     .map((result) => result.transcript)
23     .join("");
24     message.textContent = transcript;
25 };
26 recognition.onerror = function (event) {
27     message.textContent = "Error occurred in recognition: " + event.error;
28     recognizing = false;
29 };
30
31 const copyPaseText = () => {
32     var copyText = document.querySelector("#note").value;
33     navigator.clipboard.writeText(copyText).then(
34         function () {
35             alert(" Copying to clipboard was successful!");
36         },
37         function (err) {
38             console.error("Async: Could not copy text: ", err);
39         }
40     );
41 };
42 const clearNote = () => {
43     window.location.reload();
44 };
45
46 micStart.addEventListener("click", () => {
47     recognition.start();
48 });
49 copy__Text.addEventListener("click", function () {
50     copyPaseText();
51 });
52 clear__Note.addEventListener("click", function () {
53     clearNote();
54 });
```

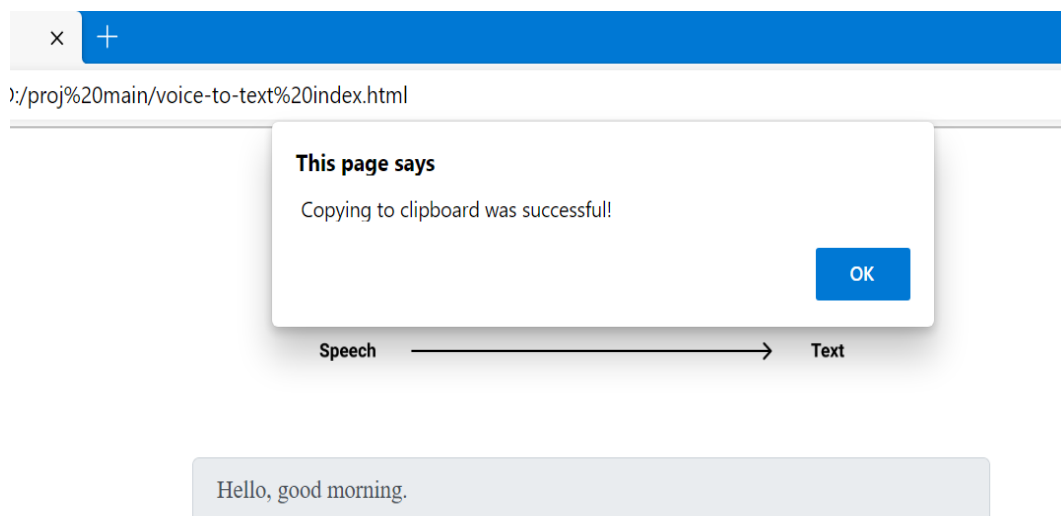
Line 1, Column 1 — 54 Lines    INS    UTF-8    JavaScript    Spaces: 4

**Fig.42.**coding part of js-2

- In Voice to Text we allocate the microphone option when users click on it. It will ask user to allow as shown in figure 22 after allowing then recording will be start. If due to any issue users wants to stop the recording he/she can stop it by clicking delete option (red color) and at same time user can copy the text till it by clicking on copy option as shown in figure 23.



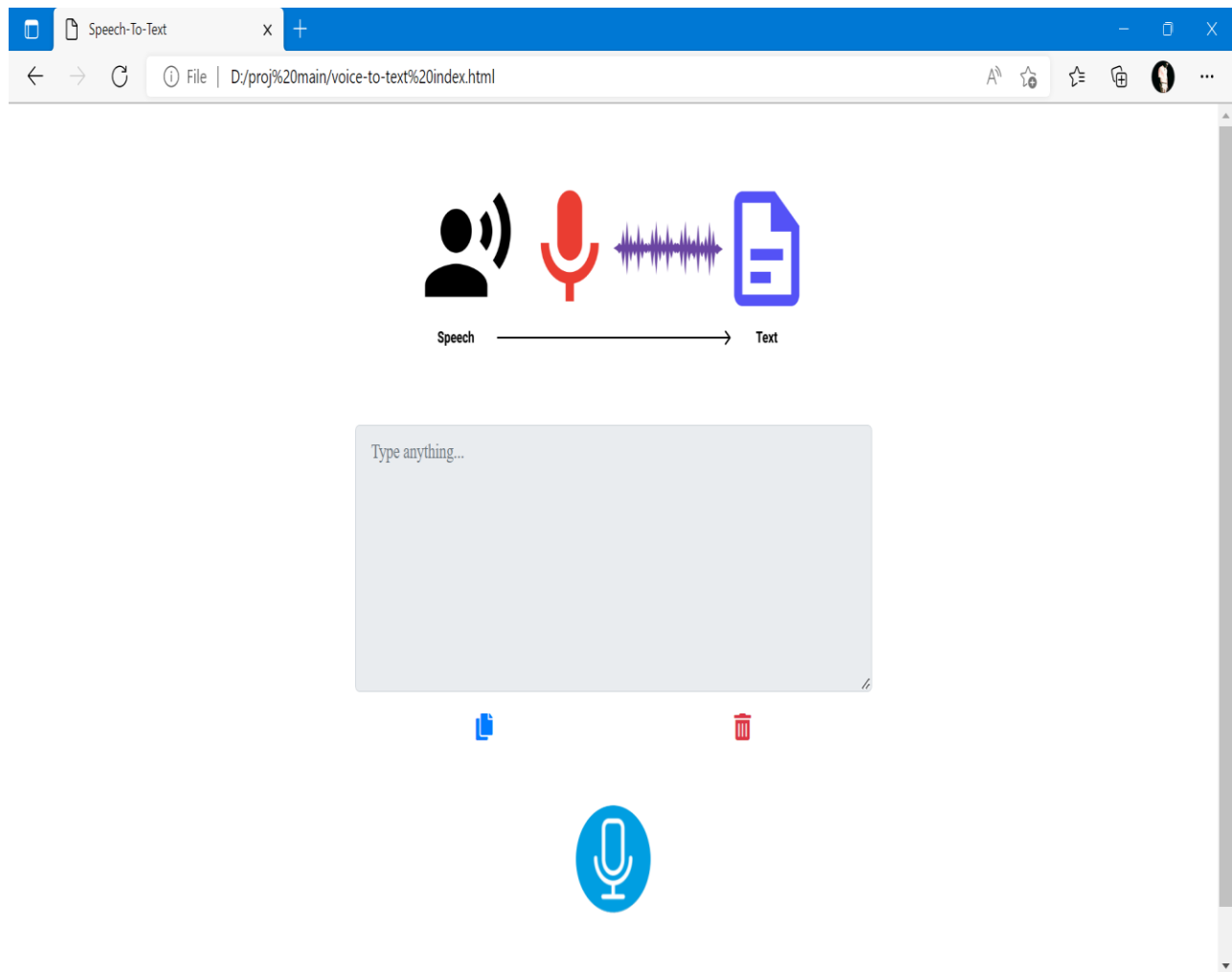
**Fig.43.**allow microphone



**Fig.44.**copying text



- Final output of Voice to Text.



**Fig.45.** final output of VTT

## 6 CONCLUSIONS

We have learned about various techniques that fall under VTT and TTS, and have also read about their applications and usage. After having looked upon closely, at the different types of speech, speech recognition, speech to text conversion, text to speech conversion and speech translation systems, we can draw a conclusion as such: In VTT, under the 2 studied we can say that API works as a better speech signal to text converter despite its drawbacks because of their computational feasibility. Similarly under TTS systems studied formant synthesis that makes use of parallel and cascade synthesis works as the best converter. Speech Synthesis translation is widely used due to its inculcation of advantages of both rule- based as well as statistical machine translation techniques. It makes sure that there is a creation of syntactically connected and grammatically correct text while also taking care of smoothness in a text, fast learning ability, data acquisition.

## 7 REFERENCES

- [1] Pratik K. Kurzekar, Ratnadeep R. Deshmukh, Vishal B. Waghmare, Pukhraj P. Shrishrimal, A Comparative Study of Feature Extraction Techniques for Speech Recognition System, International Journal of Innovative Research in Science, Engineering and Technology (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 12, December 2014.
- [2] Ms. Anuja Jadhav, Prof. Arvind Patil, Real Time Speech to Text Converter for Mobile Users, National Conference on Innovative Paradigms in Engineering Technology (NCIPET-2012) Proceedings published by International Journal of Computer Applications (IJCA)
- [3] Suman K. Saksamudre, P.P. Shrishrimal, R.R. Deshmukh, A Review on Different Approaches for Speech Recognition System, International Journal of Computer Applications (0975 8887) Volume 115 No. 22, April 2015.
- [4] Sunanda Mendiratta, Dr. Neelam Turk, Dr. Dipali Bansal, Speech Recognition by Cuckoo Search Optimization based Artificial Neural Network Classifier, 2015 International Conference on Soft Computing Techniques and Implementations- (ICSCTI) Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015.
- [5] Pere Pujol Marsal, Susagna Pol Font, Astrid Hagen, H. Bourlard, and C. Nadeu, Comparison And Combination Of Rasta-Plp And Ff Features In A Hybrid Hmm/Mlp Speech Recognition System, Speech and Audio Processing, IEEE Transactions on Vol.13, Issue: 1, 20 December 2004.
- [6] Suhas R. Mache, Manasi R. Baheti, C. Namrata Mahender, Review on Text-To-Speech Synthesizer, International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 8, August 2015.
- [7] G. E. Dahl, D. Yu, L. Deng, A. Acero, Large vocabulary continuous speech recognition with context-dependent DBN-HMMs, In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4688-4691, 2011.

- [8] Aditi Kalyani, Priti S. Sajja, A Review of Machine Translation Systems in India and different Translation Evaluation Methodologies, International Journal of Computer Applications (0975 8887) Volume 121 No.23, July 2015
  
- [9] Mouiad Fadiel Alawneh, Tengku Mohd Sembok Rule-Based and Example-Based Machine Translation from English to Arabic, 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications
  
- [10] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Speech Synthesis Based on Hidden Markov Models, Proceedings of the IEEE — Vol. 101, No. 5, May 2013. Junichi Yamagishi, Member IEEE, and Keiichiro Oura
  
- [11] F. Seide, G. Li, D. Yu, Conversational Speech Transcription Using Context-Dependent Deep Neural Networks, In Interspeech, pp. 437440, 2011.
  
- [12] Kamini Malhotra, Anu Khosla, Automatic Identification of Gender Accent in Spoken Hindi Utterances with Regional Indian Accents, 978-1-4244-3472-5/08/25.00 2008 IEEE
  
- [13] Om Prakash Prabhakar, Navneet Kumar Sahu, "A Survey on Voice Command Recognition Technique" International Journal of Advanced Research in Computer and Software Engineering, Vol 3, Issue 5, May 2013.
  
- [14] Tatsuhiko KINJO, Keiichi FUNAKI, "ON HMM SPEECH RECOGNITION BASED ON COMPLEX SPEECH ANALYSIS", 1-4244-0136-4/06/20.00 '2006 IEEE
  
- [15] Mathias De Wachter, Mike Matton, Kris Demuynck, Patrick Wambacq, Template Based Continuous Speech Recognition, IEEE Trans. On Audio, Speech Language Processing, vol.15, issue 4, pp
  
- [16] Lawrence Rabiner, Biing-Hwang Juang, B. Yegnanarayana, Fundamentals of Speech Recognition.

- [17] Jingdong Chen, Member, Yiteng (Arden) Huang, Qi Li, Kuldip K. Paliwal, "Recognition of Noisy Speech using Dynamic Spectral Subband Centroids" IEEE SIGNAL PROCESSING LETTERS, Vol. 11, Number 2, February 2004.
- [18] Hakan Erdogan, Ruhi Sarikaya, Yuqing Gao, "Using semantic analysis to improve speech recognition performance" Computer Speech and Language, ELSEVIER 2005.

### **GitHub Link:**

<https://github.com/vineethsai239/Text-To-Speech-and-Voice-To-Text.git>