**Brief Overview**

Calculator has always been the simplest and most important tool for carrying out basic as well as scientific calculations. Earlier, we commonly knew them as devices operating on batteries, solar power etc but with the increasing slope in technology and advancements in Computer Science they appeared in digital form, offering numerous features ranging from basic to scientific in a single application.

In this project, we have tried to take this a step further by introducing in it the function of ‘**Voice Recognition & Processing’**. It allows the user to give commands through speech which is received by the calculator which then processes the input and gives the result in accordance to it (an application of **‘Natural Language Processing’**).

The process involves:

1. Recording & Training of the system
2. Programming the application
3. Integrating the program with the Voice recognizing features

Along with these, the calculator also has a unique feature of ‘**Narration’**, thus also extending the accessibility options. This tool has been especially included for the easy use and help of the **‘Visually Impaired People’.**

**Softwares & Tools used**

1. Java Programming Language
2. CMU Sphinx-4 (speech recognition system written entirely in the Java(TM) programming language)
3. Eclipse mars ( IDE for Java )
4. JSpeech Grammar Format
5. Sphinx Dictionary (as Database)

**About Sphinx**

Sphinx-4 is a speech recognition system written entirely in the Java(TM) programming language.

CMU Sphinx, also called Sphinx in short, is the general term to describe a group of [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition) systems developed at [Carnegie Mellon University](https://en.wikipedia.org/wiki/Carnegie_Mellon_University). These include a series of speech recognizers (Sphinx 2 - 4) and an [acoustic model](https://en.wikipedia.org/wiki/Acoustic_Model) trainer (SphinxTrain)

Sphinx 4 is a complete re-write of the Sphinx engine with the goal of providing a more flexible framework for research in speech recognition, written entirely in the Java programming language. [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) supported the development of Sphinx 4 and contributed software engineering expertise to the project. Participants included individuals at MERL, [MIT](https://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology) and [CMU](https://en.wikipedia.org/wiki/Carnegie_Mellon_University).

Current development goals include:

1. developing a new (acoustic model) trainer
2. implementing speaker adaptation (e.g. MLLR)
3. improving configuration management
4. creating a [graph-based UI](https://en.wikipedia.org/wiki/ConfDesigner) for graphical system design

**JSpeech Grammar Format**

The JSpeech Grammar Format (JSGF) is a platform independent, Vendor independent textual representation of grammars for use in speech recognition. Grammars are used by speech recognizers to determine what the recognizer should listen for, and so describe the utterances a user may say. JSGF adopts the style and conventions of the JavaTM Programming Language in addition to use of traditional grammar notations.

Speech recognition systems provide computers with the ability to listen to user speech and determine what is said. Current technology does not yet support unconstrained speech recognition: the ability to listen to any speech in any context and transcribe it accurately. To achieve reasonable recognition accuracy and response time, current speech recognizers constrain what they listen for by using grammars ( which is more accurate but has a limited domain) & statistics ( which is comparatively less accurate but has a broader domain).

The JSpeech Grammar Format (JSGF) is a way of describing one type of grammar, a rule grammar (also known as a command and control grammar or regular grammar). It uses a textual

representation that is readable and editable by both developers and computers, and can be included in source code. The other major grammar type, the dictation grammar, is not discussed in this document.

A rule grammar specifies the types of utterances a user might say (a spoken utterance is similar to a written sentence). For example, a simple window control grammar might listen for "open a file", "close the window", and similar commands. What the user can say depends upon the context: is the user controlling an email application, reading a credit card number, or selecting a font? Applications know the context, so applications are responsible for providing a speech recognizer with appropriate grammars.

**Natural Language Processing**

Natural language processing (NLP) is a field of [computer science](https://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), and [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) concerned with the interactions between [computers](https://en.wikipedia.org/wiki/Computer) and [human (natural) languages](https://en.wikipedia.org/wiki/Natural_language). As such, NLP is related to the area of [human–computer interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction). Many challenges in NLP involve: [natural language understanding](https://en.wikipedia.org/wiki/Natural_language_understanding), enabling computers to derive meaning from human or natural language input; and others involve [natural language generation](https://en.wikipedia.org/wiki/Natural_language_generation).

**Speech Recognition**

Speech recognition (SR) is the [inter-disciplinary](https://en.wikipedia.org/wiki/Inter-disciplinary) sub-field of [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) which incorporates knowledge and research in the [linguistics](https://en.wikipedia.org/wiki/Linguistics), [computer science](https://en.wikipedia.org/wiki/Computer_science), and [electrical engineering](https://en.wikipedia.org/wiki/Electrical_engineering) fields to develop methodologies and technologies that enables the recognition and [translation](https://en.wikipedia.org/wiki/Translation) of spoken language into text by computers and computerized devices such as those categorized as smart technologies and [robotics](https://en.wikipedia.org/wiki/Robotics). It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT).

Some SR systems use "training" (also called "enrollment") where an individual speaker reads text or isolated [vocabulary](https://en.wikipedia.org/wiki/Vocabulary) into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. There are basically two types of systems namely, ‘Speaker Dependent Systems’ and ‘Speaker Independent Systems’.

Speech recognition applications include [voice user interfaces](https://en.wikipedia.org/wiki/Voice_user_interface) such as voice dialing (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), [domotic](https://en.wikipedia.org/wiki/Domotic" \o "Domotic) appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., [word processors](https://en.wikipedia.org/wiki/Word_processor) or [emails](https://en.wikipedia.org/wiki/Email)), and [aircraft](https://en.wikipedia.org/wiki/Aircraft) (usually termed [Direct Voice Input](https://en.wikipedia.org/wiki/Direct_Voice_Input)).

The term speaker identification refers to identifying the speaker, rather than what they are saying. [Recognizing the speaker](https://en.wikipedia.org/wiki/Speaker_recognition) can simplify the task of translating speech in systems that have been trained on a specific person's voice or it can be used to authenticate or verify the identity of a speaker as part of a security process.

From the technology perspective, speech recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in [deep learning](https://en.wikipedia.org/wiki/Deep_learning) and [big data](https://en.wikipedia.org/wiki/Big_data). The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the world-wide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems. These speech industry players include [Google](https://en.wikipedia.org/wiki/Google), [Microsoft](https://en.wikipedia.org/wiki/Microsoft), [Hewlett Packard Enterprise](https://en.wikipedia.org/wiki/Hewlett_Packard_Enterprise), [IBM](https://en.wikipedia.org/wiki/IBM), [Baidu](https://en.wikipedia.org/wiki/Baidu" \o "Baidu) (China), [Apple](https://en.wikipedia.org/wiki/Apple_Inc.), [Amazon](https://en.wikipedia.org/wiki/Amazon.com), [Nuance](https://en.wikipedia.org/wiki/Nuance_Communications), [IflyTek](https://en.wikipedia.org/wiki/Anhui_USTC_iFlytek_Co." \o "Anhui USTC iFlytek Co.) (China), many of which have publicized the core technology in their speech recognition systems as being based on deep learning.

**The 3 Vital Files**

The whole process involves the creation of 3 types of files without which the application cannot work -

1. Configuration file
2. Grammar File
3. Program File

**Configuration File:**

The configuration of a particular Sphinx-4 system is determined by a configuration file. This configuration file defines the following:

* The names and types  of all of the components of the system
* The connectivity of these components - that is, which components talk to each other
* The detailed configuration for each of these components.

Some things to note about this configuration file :

* The format of the file is XML
* This configuration file defines a single component called mySampleComponent.
* The type of this component is edu.cmu.sphinx.sample.MyComponent which must implement the [Configurable](http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/util/props/Configurable.html) interface.

**Grammar File:**

A grammar is composed of a set of rules that together define what may be spoken.

Rules are combinations of speakable text and references to other rules. Each rule has a unique rulename. A reference to a rule is represented by the rule's name in surrounding <> characters (less than and greater than).

A single file defines a single grammar. The definition grammar contains two parts: the grammar header and the grammar body. The grammar header includes a self identifying header, declares the name of the grammar and declares imports of rules from other grammars. The body defines the rules of the grammar, some of which may be public.

The grammar body defines rules. Each rule is defined in a rule definition. The components of the rule definition are:

(1) An optional public declaration,

(2) The name of the rule being defined,

(3) An equals sign `=',

(4) The expansion of the rule, and

(5) A closing semicolon `;'.

**Program File**

The program file is the main java file where all the code is written for the application. It is here where it is created, from giving it a frame to making it look attractive; assigning all the functions which includes accepting input from the user, processing it and finally providing the desired output.

This file also includes the link for the configuration file which acts as a bridge for all other tools which are required in the program.

**How it works?**

The Voice Enabled Calculator works on a series of steps for processing the voice input and giving the output. All the above mentioned files- Configuration, Grammar & Program are mandatory for its working.

Following are the points encompassing a brief explanation for functioning of the application:

**Voice Recognition:**

1. An input is given by the user in the form of speech through a microphone attached to the system.
2. This voice input is recognized by of Sphinx – 4 system. This is done with the help of dictionary file of Sphinx that contains words along with their specified pronunciations. So whenever something is spoken features are extracted and matched with the features of pre-recorded pronunciations and hence the corresponding word or letter is identified.
3. Now if the recognized form of speech is present in the Grammar File of the application then only it is sent to further program otherwise it is considered as out of vocabulary words by the system. Hence the Grammar File allows us to choose specific words from the dictionary and use only them so as to increase the level of accuracy and speed.
4. In this way **speech-to-text conversion** takes place and the converted text can then be further used in programming, wherever required.

**Narration:**

1. Narration is the process in which the system orally dictates about any text being displayed on the screen or activity performed by the user. This technique extends the accessibility of the application.
2. This has been made possible by using the sound classes of Java. One of its methods takes an audio file as an input (but it has to possess ‘.wav’ extension) and plays it, as & when specified in the program.
3. The main task is to do the segmentation of the text being displayed on the screen and then commanding the system to play the sound in accordance to it. Hence, of course for this we need to have all the required audio files to be present on our machine.

**Few Illustrations:**

