SignIT – Audio to ASL Translator

Department of Software Engineering

Capstone Project Phase 1, 21-2-D-3

Supervisor:

Mr. Alexander Keselman

Authors:

Ido Kadosh IdoKadosh1@gmail.com

Itay Ziv ItayZiv8@gmail.com

Table of Contents

Abstract

A majority of deaf 18-years-olds in the United States have an English reading level bellow that of a typical 10-year-old student, and so machine translation software that could translate English into American Sign Language (ASL) could significantly improve these individuals' access to information, communication and services. This project tries to offer a solution to this problem by using techniques such as NLP and Speech Recognition to allow the deaf community the ability to improve those aspects. Another feature this project offers are the ability to communicate between two people where one of them is deaf, and another is not familiar with the ASL, without a middle-man. In order to achieve those goals this project allows its users to record a real-time recording in the English language and get the ASL translation by seconds.

1. Introduction

Hand gestures are used as a way for people to express thoughts and feelings, it serves reinforce information delivered in our daily conversation. Sign language is a structured form of hand gestures involving visual motions and signs, which are used as a communication system. For the deaf and speech-impaired community, sign language serves as useful tools for daily interaction.

However, sign language is not common among the hearing community, and fewer are able to understand it. This poses a genuine communication barrier between the deaf community and the rest of the society, as a problem yet to be fully solved until this day.

A sign language consists of a well-structured code of signs, and gestures, each of which has a particular meaning assigned to it. They have their own grammar and lexicon. It includes a mixture of hand positioning, shapes and movements of the hands.

The need of this project stems from the desire to integrate the deaf and speech-impaired community, into the rest of the world population without the need for an interpreter or any other intermediate.

This software aims to get the deaf and speech-impaired community more involved to communicate, and the idea of an audio conversion into sign language system that would be in use for converting an existing audio files or a real time audio files such as recorded messages to sign language using sequence of videos.

This project goal is to design a solution that is intuitive and simple which simplifies the communication for the majority of people with deaf and dumb'.
Sign language translation software will be able to improve communication and allow the deaf community to enjoy full participation in day-to-day interaction and access more information and services.
Moreover, the system will not only improve information access, but it can also be used as an educational tool for American sign language education, so in the future this problem will not be even exist and the deaf and speech-impaired community will be an integral part of the world population without any barriers.

1.1 Paper Overview

The rest of the paper is organized as follow. Section 2 displays the background and the related work done so far in order to solve the problem we presented.
In section 3, we emphasize our goals and our expected achievements.
Section 4 devoted to the research process we performed during the semester in order to achieve the goals we mentioned, and the motivation for it. In addition, this section detailed the constraints that may affect our development process.
In section 5, we describe our product, this section includes explanation about our software and our testing methodology.
At last, the evaluation metric used in this work are showcased in section 6.

2. Background and Related Work

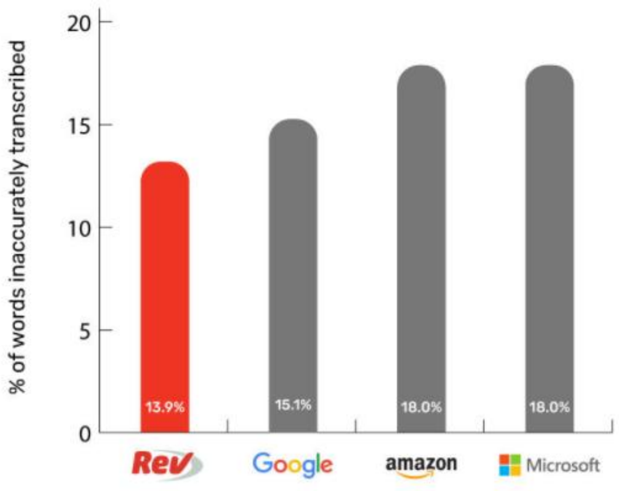
American Sign language (ASL) is the natural language of around 500,000 deaf people in the US and Canada. The National Center for Health Statistics estimates that 28 million Americans (about 10% of the population) have some degree of hearing loss. About 2 million of these 28 million are classified as deaf.

The most common misconception about ASL is that it is a signed version of English. ASL is not English at all. ASL is a distinct language with its own syntax and grammar and has been developed over hundreds of years by deaf people as a means of communication.

This section displays work which will contribute to the implementation phase of this project.

2.1 Google SpeechRecognition API

Recognizing speech requires audio input, and SpeechRecognition makes retrieving this input really easy. Instead of having to build scripts for accessing microphones and processing audio file from scratch. The SpeechRecognition library acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these – the Google Web Speech API – supports a default API key that is hard-coded into the SpeechRecognition library.



2.2 WordNet

WordNet was first created in English only in the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George Armitage Miller starting in 1985 and has been directed in recent years by Christiane Fellbaum.

WordNet is a Natural Language Processing (NLP) resource that is a lexical database for the English language. WordNet is made up of groups or set of words that are synonyms of each other. Each of these sets is called a synset. Since a word can have more than one meaning it can belong to more than one synset. Using WordNet gives the option to find not only a word's synonyms, but also, it's hyponyms, hypernyms, holonyms, meronyms and antonyms.

The database contains 155,287 words organized in 117,659 synsets for a total of 206,941 word-sense pairs.

2.3 SigningSavvy

SigningSavvy was created by Jillian Winn and John Miller in order to educate people the American Sign Language.

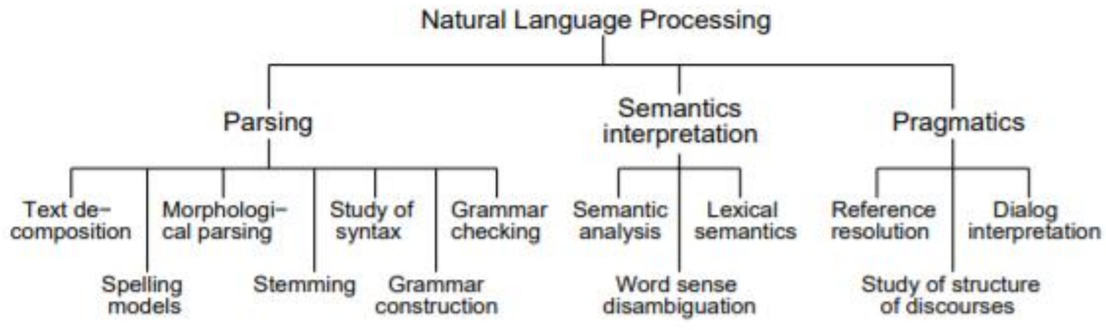
SigningSavvy contains high resolution videos of American Sign Language (ASL) signs and common other signs used in conversational signing with the United States and Canada. This website has videos of signs for more than 7000 words and phrases. Any word not contained in the website can be fingerspelled.

2.4 Natural Language Processing (NLP)

NLP refers to the branch of computer science and more specifically, the branch of Artificial intelligence – concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics – rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning.

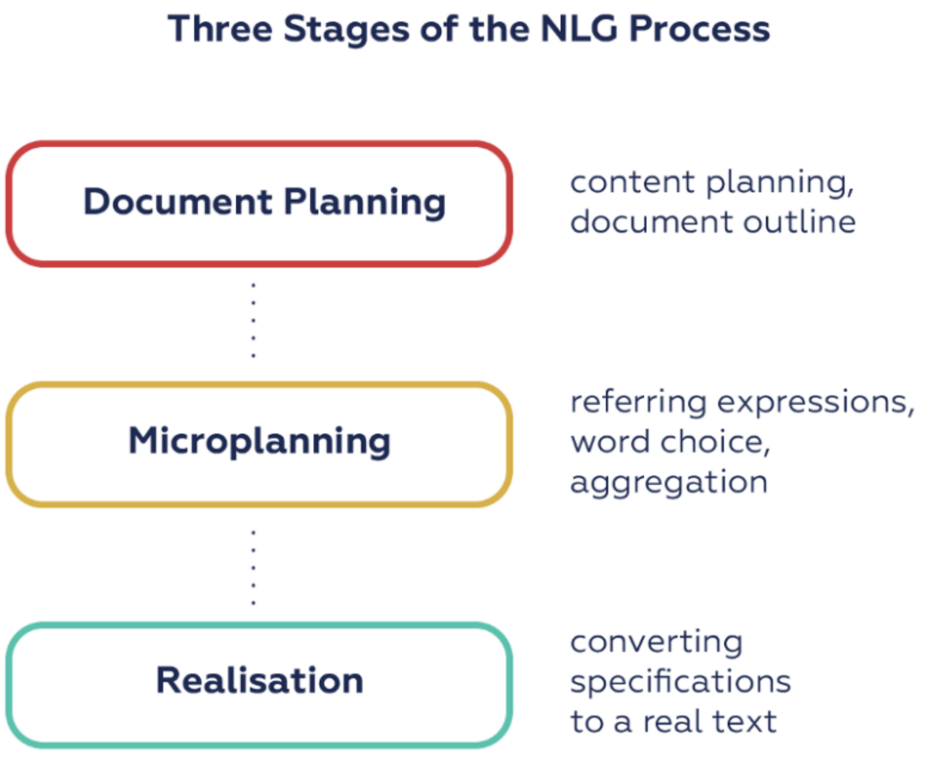
NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly (even in real time).



2.4.1 *Natural Language Generation (NLG)*

NLG, a subfield of artificial intelligence, is a software process that automatically transforms data into plain English content. NLG is one of the fastest growing technologies being adopted in the enterprise.

Many NLG systems consist of 3 components which are connected in a pipeline. i.e., the output of document planning acts as input to microplanning and the output of the micro planner is the input to the surface realizer.


Three Stages of the NLG Process
Document Planning
content planning, document outline
Microplanning
referring expressions, word choice, aggregation
Realisation
converting specifications to a real text

Three Stages of the NLG Process

2.4.1.1 *SimpleNLG*

SimpleNLG is an open-source Java API designed to facilitate the generation of Natural Language. It was originally developed by Ehud Reiter, Professor at the University of Aberdeen's Department of Computing Science and co-founder of Arria NLG.

SimpleNLG can be used to write a program which generates grammatically correct English sentences, this library performs simple and useful tasks that are necessary for NLG.

2.5 *ASL Linguistics*

ASL is a language with its own structure, form and grammar that separates it from English. ASL Linguistics is the study of this language and the rules that govern it. The purpose of this section is to display the linguistics of ASL.

2.5.1 ASL Morphology

Morphology is a field of linguistics focused on the study of the forms and formation words in a language. Words in language consist of one element or elements of meaning which are called morphemes.

The smallest meaningful unit in a language is called a morpheme. In this study of ASL Linguistics there are two different morphology processes called derivational morphology and inflectional morphology.

Derivational morphology is the process of making new units for the language by adding affixes. Such is the case when creating a noun from a verb. For example, adding the suffix "-r" to the end of the verb "write" to create the noun "writer".

Inflectional Morphology is the process of adding grammatical information to units that already exist. For example, changing the word "look" to "looks" or "looking" by adding the suffixes "-s" and "-ing".

2.5.2 ASL Phonology

Phonology is the study of how signs are structured and organized. There are five basic parts to ASL signs – Handshape, Movement, Location, Orientation, Nonmanual signs or facial expression. There are many signs that will have the same parameters for handshape, movement and location.

Sign languages such as ASL are characterized by phonological analogous to, yet dissimilar from, those of oral languages. Although there is a qualitative difference from oral languages in that sign-language phonemes are not based on sound, and are spatial in addition to being temporal, they fulfill the same role as phonemes in oral languages.

Basically, three types of signs are distinguished: one-handed signs, symmetric two-handed signs, and asymmetric two-handed signs. The non-dominant hand in asymmetric signs often functions as the location of the sign. Almost all simple signs in ASL are monosyllabic.

2.5.3 ASL Syntax

Syntax is the combining of words and signs to create phrases and sentences. There are six basic sentence types: questions, negations, commands, topicalization, conditionals and declarative.

Word Order - In English, the order of words is very important for the meaning of the sign. However, in ASL it is possible to change the word order and yet obtain the same meaning. This is frequently done through topicalization and nonmanual signs.

Time and Aspect – In ASL, tense is represented by the use of space. For example, the signer's body is used as a reference for the present. Leaning back or referring to behind the signer is used to represent the past. Conversely, leaning forward and signing in front of the signer is used to represent the future.

Aspect, in ASL, uses specific movements to represent aspects such as: over and over, for a prolonged period of time, continually, regularly, etc.

Indicating Verbs – There are three main categories of verbs in ASL: plain, indicating and depicting.

2.5.4 ASL Semantics

Semantics is the study of the meaning of words and sentences. In order for the morphology, phonology and syntax of a language to be used for communication there needs to be a shared system of meaning.

Relationships between Lexical Items – Hyponymy is relationship between a blanket term and other terms that are included in it. For example, COLOR will include, WHITE, BLUE, RED, etc. Synonymy is the relationship between words with the same meaning. For example, DAD and FATHER. Antonymy is the relationship between words that have opposite meaning.

The Meaning of Sentences – Sentences can mean different things depending on context. It is important to know the context or background of a conversation or else meaning of a sentence can be confused.

3. Expected Achievements

This project intended to provide a solution to the main problem of the deaf and the speech-impaired community and that is the ability to communicate with the rest of the population who do not familiar with their language – Sign Language. Other purpose of this project is to provide a software that will serve as a learning tool for American Sign Language – ASL.

The outcome of this project will be a conversion software that will translate existing audio files in any audio format given, or a real-time recording using the computer/smartphone microphone to a sequence of videos that will display a grammatically correct sentence by using "SigningSavvy".

This project main criteria for success are to provide a correct translation from the English language to ASL without compromising the context of the sentence.

4. Research/Engineering Process

This section includes the research process we have conducted. This section describes the work process we performed during the semester in order to accomplish the project goal. This section is divided into two parts, the process and the product of the project.

4.1 Process

In this project, we face the problem of translating English to American Sign Language (ASL). We began our engineering process by researching existing technologies and concepts that will help us deal with this problem.

In order to accomplish this project goals, we went over the related background for those methods as described in section 2. We have conducted research on a wide range of technologies such as Speech Recognition and Natural Processing Language (NLP). We also needed to conduct research about the ASL Linguistics in order to ensure correct translation. The main reason we chose these technologies is the efficiency of NLP for preserving the content of a sentence and separate it into parts (such as verbs, adjectives, subject and etc.). Regarding using Speech Recognition library, we encountered several issues. We have found better technologies from Google Speech Recognition API, such as Rev.AI, but in order to use this technology we were required to pay money, this led us to integrate between Python and Java where the use of Speech Recognition library that implemented in Python allows us to *make the two desired transcribes in our software – a real time transcription to text, and an existing audio file transcription to text. by using Java we perform the rest of the work.*

One of the things that made it especially difficult for us in the research process is the absence of datasets/databases. We have encountered a research about database that provides translation from English to ASL wrote by Prof. Ronnie Wilbur at Purdue University, but this database is private. When we tried to reach out in order to get access, we haven't got any response. In order to overcome this major setback, we kept searching and found the 'SigningSavvy' website and decided to adopt it as our database.

The next steps in our research process will be to consider in which audio formats our software will support, of course our intent that it will support at any audio format. In order to ensure that, we will continue to analyze the options Speech Recognition allows. Another thing we have not yet researched is the option to add subtitles while our software displays the ASL translation, in order to allow another aspect of conveying the message.

4.2 Product

This section provides a high-level description for the software development process. At section 4.2.1 described the software flow sequence. In section 4.2.2 this paper describes the models that used in order to accomplish a correct translation from English to ASL. Sections 4.2.3 and 4.2.4 are presented in order to describe the dynamic aspects and concepts of the system by showing Activity Diagram and Conceptual Diagram. Section 4.2.5 shows the Sign-IT prototype and contains parts of the code that will be used in the implementation process. At last, in section 4.2.7 this paper presents the Graphical User Interface.

4.2.1 Sign-IT Workflow

The conversion process from audio to ASL contains 2 main phases, the first phase refers to the transcription from audio to English text. The second phase contains the translation from English to ASL by using Natural Language Processing.

Through Speech Recognition package, input is obtained whether it's listened to a real-time recording or by listening to an existing audio file and then transcribes it into text. In this phase, the software saves the transcription as a text file.

4.2.1.1 Translation Process

The second phase begins with the conversion process from English sentence into a grammatically correct ASL sentence. In order to ensure a correct translation, this project perform the following steps – at first, it replaces contractions in sentence with their original words. Then determines context of sentence and bring time items to the beginning of the sentence, and brings past tense items first if time items are not already. Pushes negative items to the end of the sentence. Delete be words. Get the true-form of verbs and converts any plural nouns to singular. At last, replaces the verb-adverb and adjective-noun order.

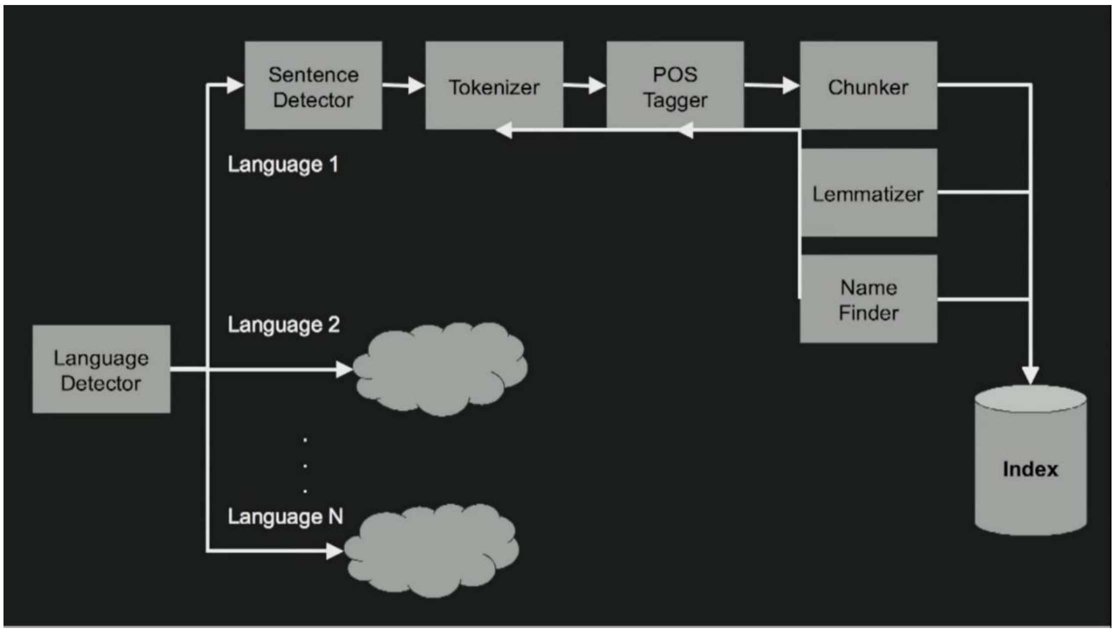
In addition, the system checks whether there is an ambiguity between words by using WordNet package [2.2].

4.2.1.2 Display ASL Translation

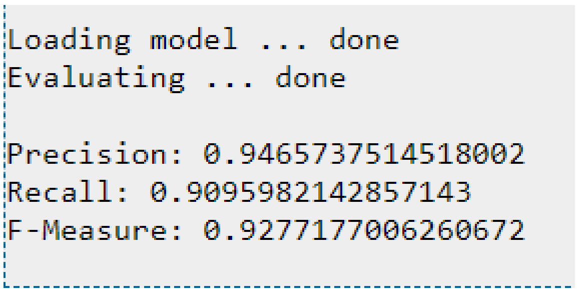
After the translation process completed, the software shows sequence of videos that represent the ASL sentence that created by the system at the previous section [4.2.1.1]. In case there is a word with several interpretations the system asks the user to clarify his intent in order to display a correct video – by displaying the options to the user, and in order to continue the user have to choose one of the options.

4.2.2 Sign-IT Models

This section describes the Apache OpenNLP models this project used in order to convert English to American Sign Language.



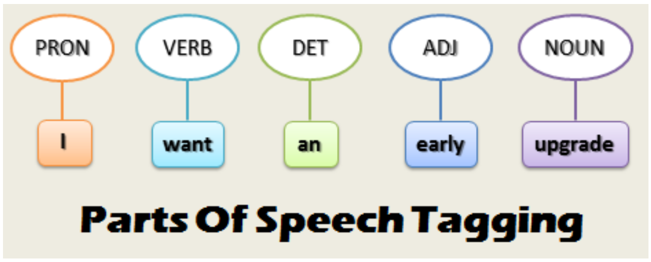
- *<u>Language Detector</u> - The OpenNLP Language Detector classifies a document in ISO-639-3 languages according to the model capabilities. To perform classification this project needs a machine learning model – these are encapsulated in the LanguageDetectorModel class of OpenNLP tools. By using this model, Sign-IT ensures that the translation refers to English and no other language.*
- *<u>Sentence Detector</u> - The OpenNLP Sentence Detector can detect that a punctuation character marks the end of a sentence or not. In this sense a sentence is defined as the longest white space trimmed character sequence between to punctuation marks, that allows this project to preserve the context of a sentence without cutting a part of it.*



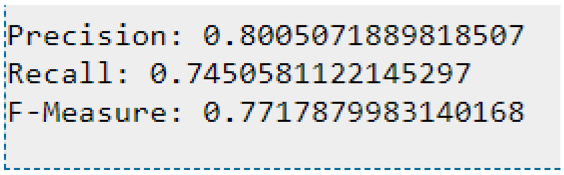
```
Loading model ... done
Evaluating ... done

Precision: 0.9465737514518002
Recall: 0.9095982142857143
F-Measure: 0.9277177006260672
```

- *Tokenizer - The OpenNLP Tokenizers segment an input character sequence into tokens. Tokens are usually words, punctuation, numbers, etc. With OpenNLP, tokenization is a two-stage process: first, sentence boundaries are identified, then tokens within each sentence are identified.*

- *Part of Speech (POS) Tagger – The POS Tagger marks tokens with their corresponding word type based the token itself and the context of the token. A token might have multiple pos tags depending on the token and the context. POS Tagger uses a probability model to predict the correct pos tag out of the tag set.*

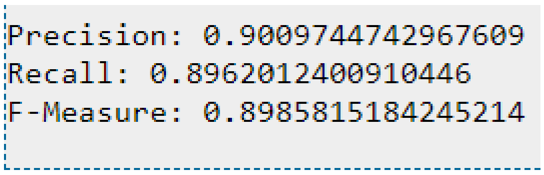


- *Chunker – Text chunking consists of dividing a text in syntactically correlated parts of words, like noun groups and verb groups, but does not specify their internal structure, nor their role in the main sentence.*
- *Lemmatizer – The Lemmatizer returns, for a given token and POS tag, the dictionary form of a word, which is usually referred to as its lemma. A token could ambiguously be derived from several basic forms or dictionary words which is why the POS tag of the word is required to find the lemma. For example : the form 'show' may refer to either the verb 'to show' or to the noun 'show'.*
- *Named Entity Recognition - The Name Finder can detect named entities and numbers in text. To be able to detect entities the Name Finder needs a model. The model is dependent on the language and entity type it was trained for. The OpenNLP projects offers a number of pre-trained name finder models which are trained on various freely available corpora. This allows Sign-IT to determine whether a word refer to a name or a place.*



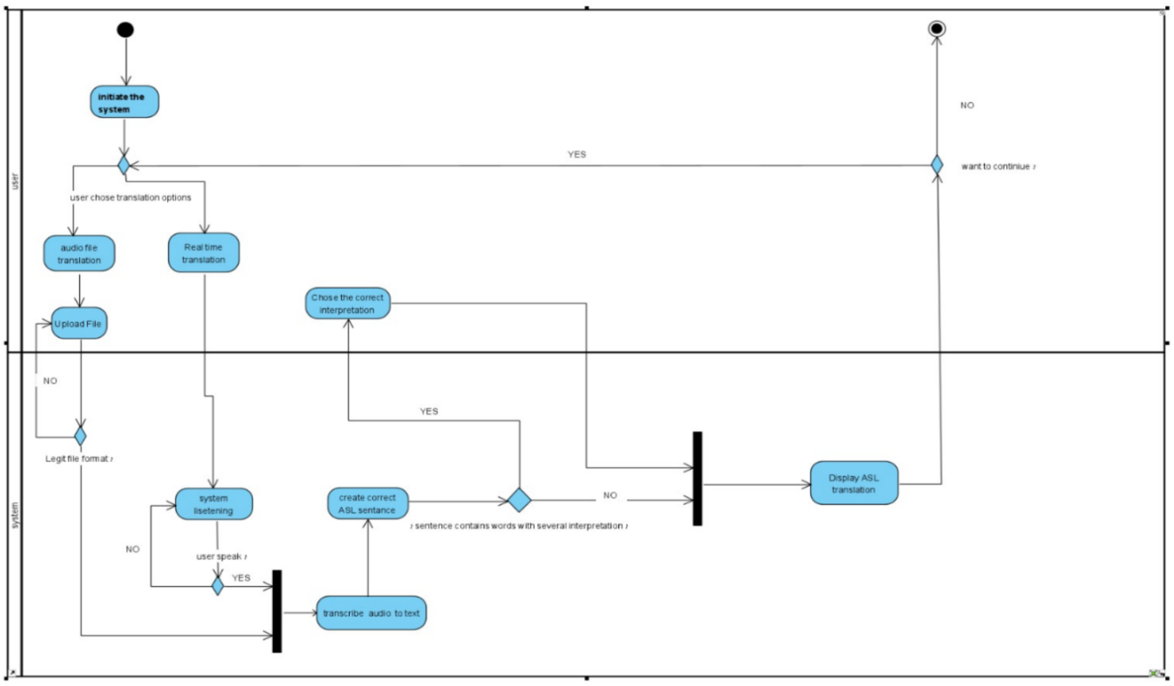
```
Precision: 0.8005071889818507
Recall: 0.7450581122145297
F-Measure: 0.7717879983140168
```

- *Parser - A parser returns a parse tree from a sentence according to a phrase structure grammar. A parse tree specifies the internal structure of a sentence. A parse tree can be used to determine the role of subtrees or constituents in the sentence, and ensures the conversion from English to ASL done correctly*

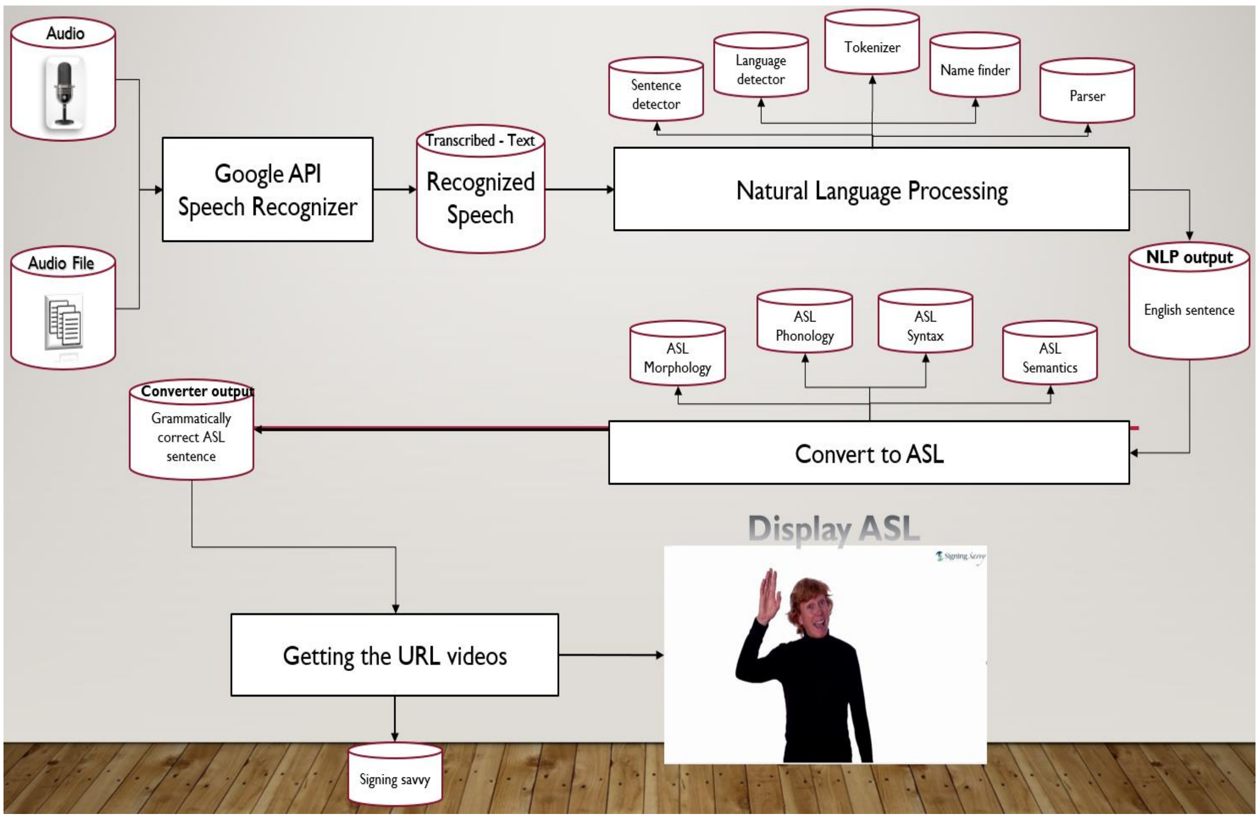


```
Precision: 0.9009744742967609
Recall: 0.8962012400910446
F-Measure: 0.8985815184245214
```

4.2.3 Activity Diagram



4.2.4 Conceptual Diagram

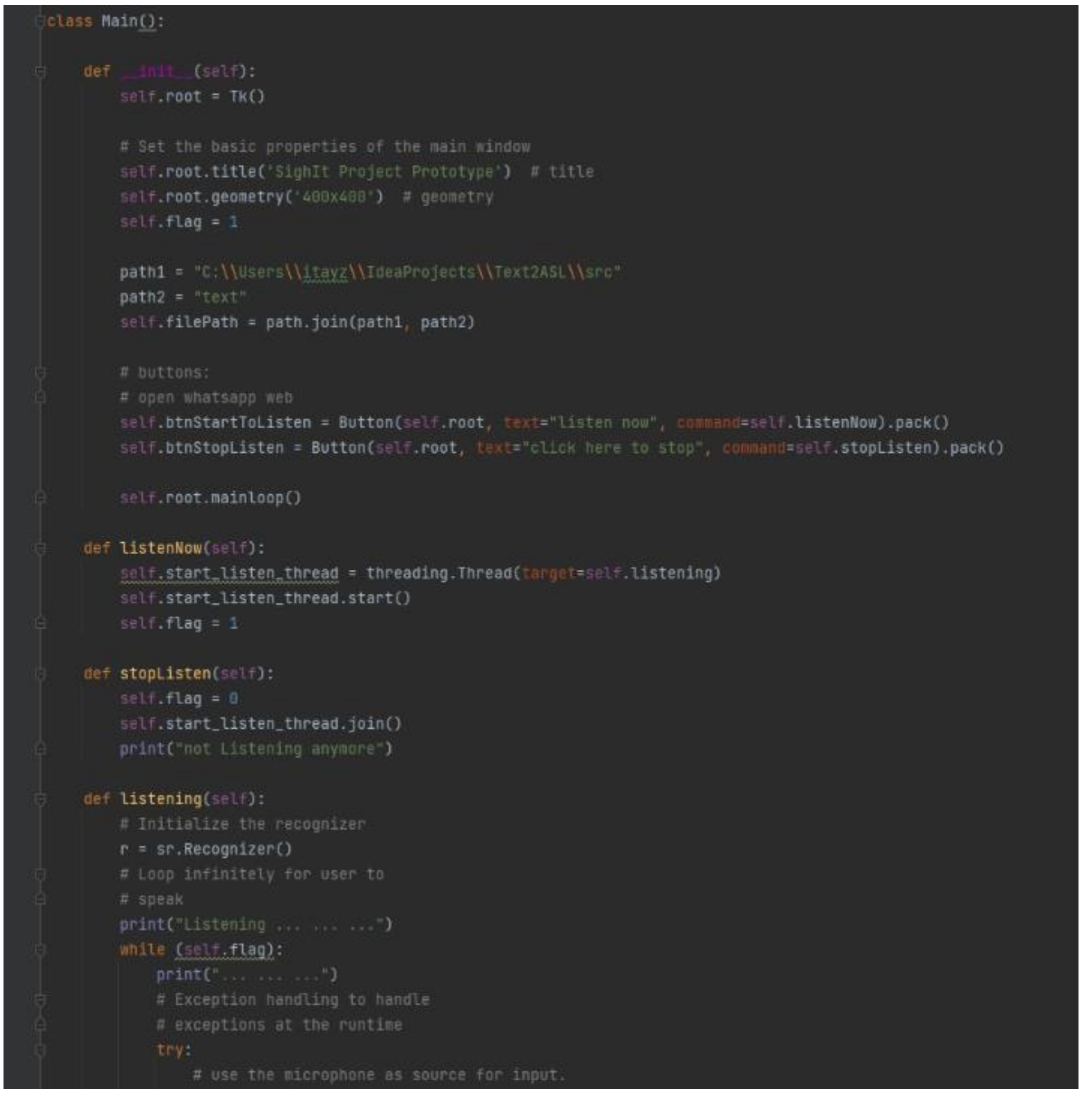


4.2.5 Prototype

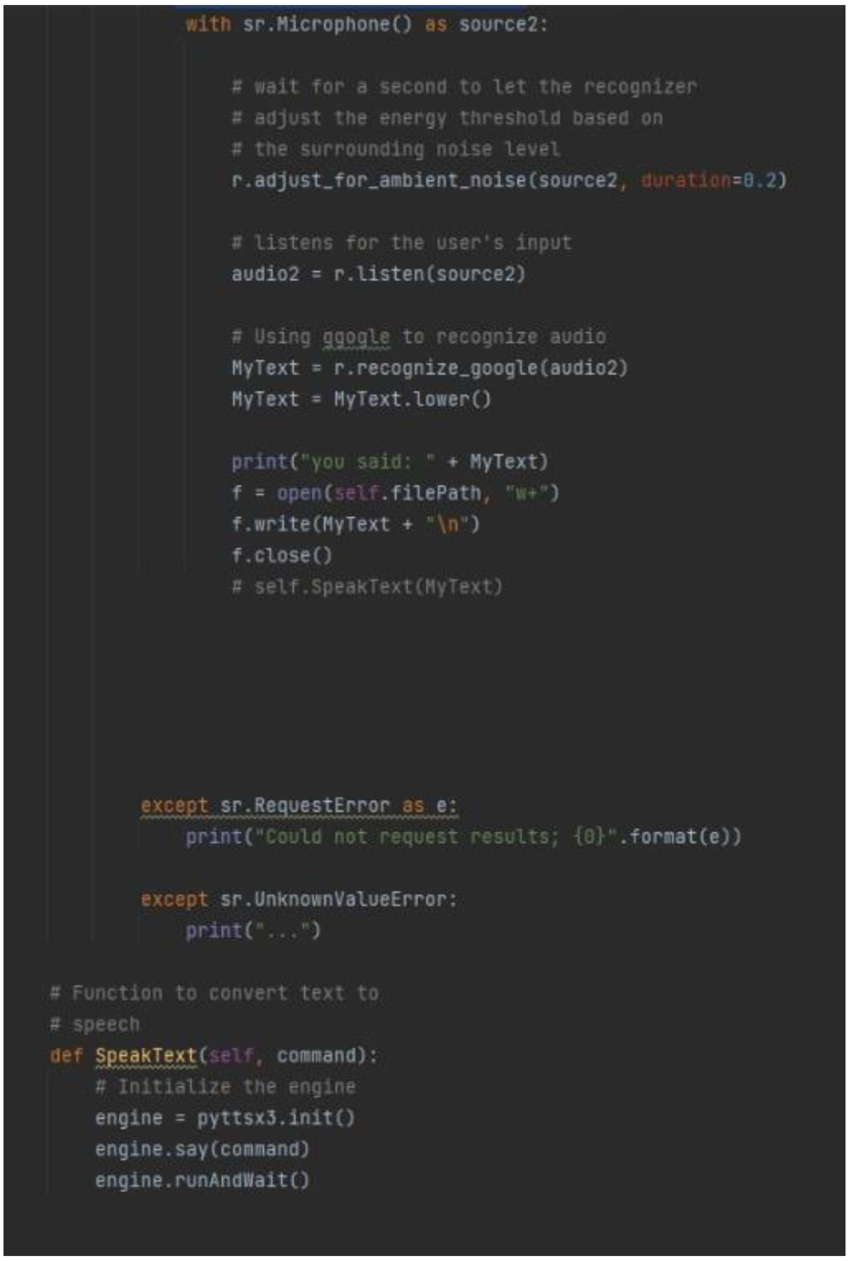
This section describes the code we already implement. We will use this code during the implementation phase.

4.2.5.1 SpeechRecognition Implementation

For now, this code used only for a real-time recording and the transcription process from an existing audio file still need to be implement.



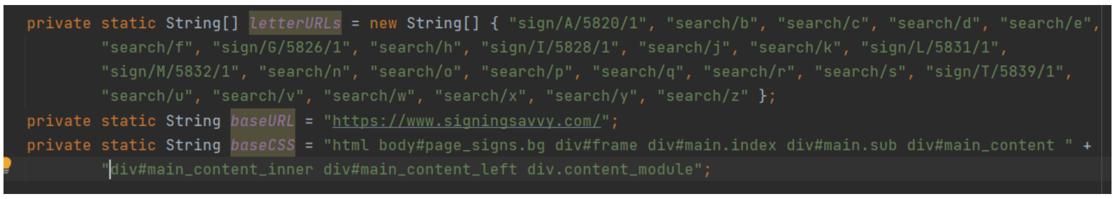
```
class Main():

    def __init__(self):
        self.root = Tk()

        # Set the basic properties of the main window
        self.root.title('SighIt Project Prototype')  # title
        self.root.geometry('400x400')  # geometry
        self.flag = 1

        path1 = "C:\\Users\\itayz\\IdeaProjects\\Text2ASL\\src"
        path2 = "text"
        self.filePath = path.join(path1, path2)

        # buttons:
        # open whatsapp web
        self.btnStartToListen = Button(self.root, text="listen now", command=self.listenNow).pack()
        self.btnStopListen = Button(self.root, text="click here to stop", command=self.stopListen).pack()

        self.root.mainloop()

    def listenNow(self):
        self.start_listen_thread = threading.Thread(target=self.listening)
        self.start_listen_thread.start()
        self.flag = 1

    def stopListen(self):
        self.flag = 0
        self.start_listen_thread.join()
        print("not Listening anymore")

    def listening(self):
        # Initialize the recognizer
        r = sr.Recognizer()
        # Loop infinitely for user to
        # speak
        print("Listening ... ... ...")
        while (self.flag):
            print("... ... ...")
            # Exception handling to handle
            # exceptions at the runtime
            try:
                # use the microphone as source for input.
```



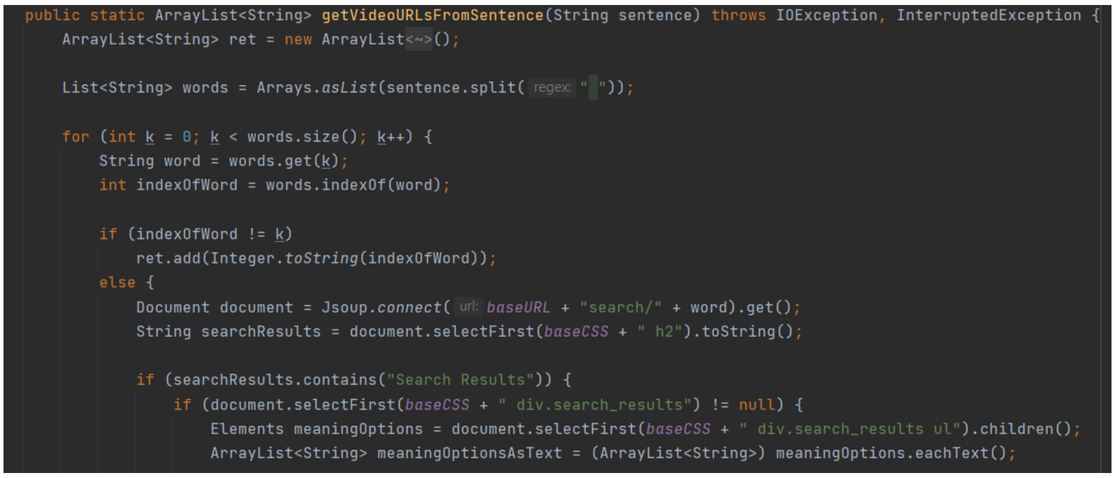
```
            with sr.Microphone() as source2:

                # wait for a second to let the recognizer
                # adjust the energy threshold based on
                # the surrounding noise level
                r.adjust_for_ambient_noise(source2, duration=0.2)

                # listens for the user's input
                audio2 = r.listen(source2)

                # Using google to recognize audio
                MyText = r.recognize_google(audio2)
                MyText = MyText.lower()

                print("you said: " + MyText)
                f = open(self.filePath, "w+")
                f.write(MyText + "\n")
                f.close()
                # self.SpeakText(MyText)




        except sr.RequestError as e:
            print("Could not request results; {0}".format(e))

        except sr.UnknownValueError:
            print("...")

# Function to convert text to
# speech
def SpeakText(self, command):
    # Initialize the engine
    engine = pyttsx3.init()
    engine.say(command)
    engine.runAndWait()
```

4.2.5.2 ASL Translation Display

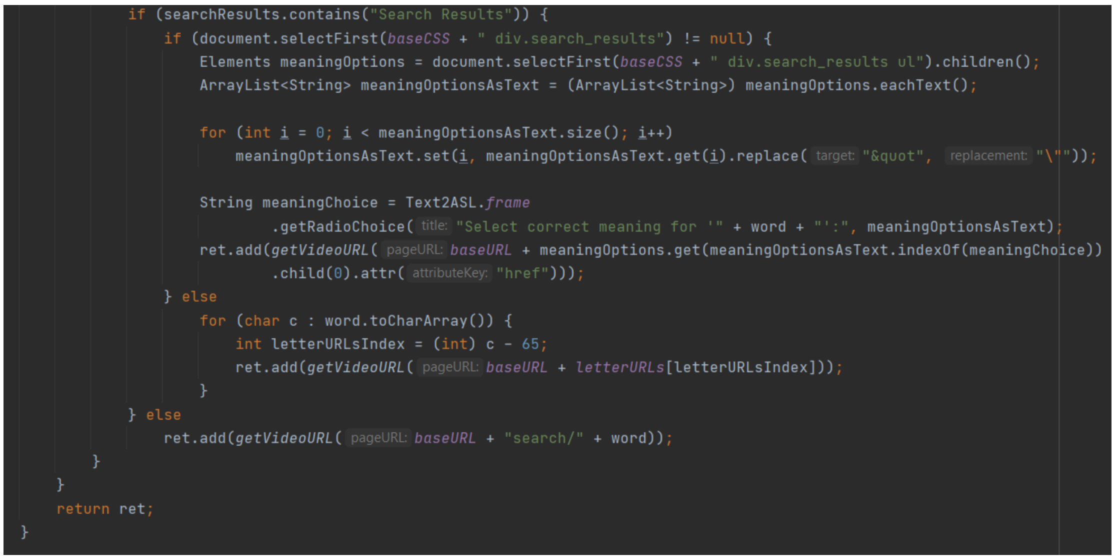
This project uses this code in order to display sequence of videos showing the ASL translation. This code is missing the subtitles showing implementation.



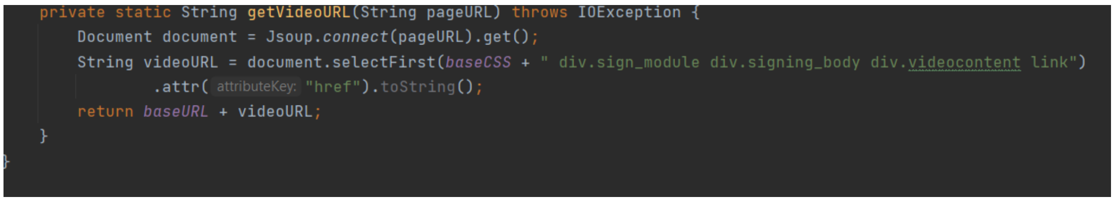
```java
private static String[] letterURLs = new String[] { "sign/A/5820/1", "search/b", "search/c", "search/d", "search/e",
        "search/f", "sign/G/5826/1", "search/h", "sign/I/5828/1", "search/j", "search/k", "sign/L/5831/1",
        "sign/M/5832/1", "search/n", "search/o", "search/p", "search/q", "search/r", "search/s", "sign/T/5839/1",
        "search/u", "search/v", "search/w", "search/x", "search/y", "search/z" };
private static String baseURL = "https://www.signingsavvy.com/";
private static String baseCSS = "html body#page_signs.bg div#frame div#main.index div#main.sub div#main_content " +
        "div#main_content_inner div#main_content_left div.content_module";
```



```java
public static ArrayList<String> getVideoURLsFromSentence(String sentence) throws IOException, InterruptedException {
    ArrayList<String> ret = new ArrayList<~>();

    List<String> words = Arrays.asList(sentence.split( regex: " "));

    for (int k = 0; k < words.size(); k++) {
        String word = words.get(k);
        int indexOfWord = words.indexOf(word);

        if (indexOfWord != k)
            ret.add(Integer.toString(indexOfWord));
        else {
            Document document = Jsoup.connect( url: baseURL + "search/" + word).get();
            String searchResults = document.selectFirst(baseCSS + " h2").toString();

            if (searchResults.contains("Search Results")) {
                if (document.selectFirst(baseCSS + " div.search_results") != null) {
                    Elements meaningOptions = document.selectFirst(baseCSS + " div.search_results ul").children();
                    ArrayList<String> meaningOptionsAsText = (ArrayList<String>) meaningOptions.eachText();
```



```java
            if (searchResults.contains("Search Results")) {
                if (document.selectFirst(baseCSS + " div.search_results") != null) {
                    Elements meaningOptions = document.selectFirst(baseCSS + " div.search_results ul").children();
                    ArrayList<String> meaningOptionsAsText = (ArrayList<String>) meaningOptions.eachText();

                    for (int i = 0; i < meaningOptionsAsText.size(); i++)
                        meaningOptionsAsText.set(i, meaningOptionsAsText.get(i).replace( target: "&quot", replacement: "\""));

                    String meaningChoice = Text2ASL.frame
                            .getRadioChoice( title: "Select correct meaning for '" + word + "':", meaningOptionsAsText);
                    ret.add(getVideoURL( pageURL: baseURL + meaningOptions.get(meaningOptionsAsText.indexOf(meaningChoice))
                            .child(0).attr( attributeKey: "href")));
                } else
                    for (char c : word.toCharArray()) {
                        int letterURLsIndex = (int) c - 65;
                        ret.add(getVideoURL( pageURL: baseURL + letterURLs[letterURLsIndex]));
                    }
            } else
                ret.add(getVideoURL( pageURL: baseURL + "search/" + word));
        }
    }
    return ret;
}
```



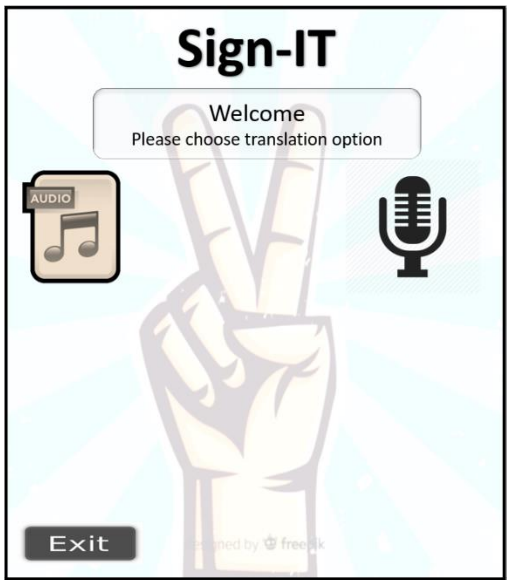
```java
    private static String getVideoURL(String pageURL) throws IOException {
        Document document = Jsoup.connect(pageURL).get();
        String videoURL = document.selectFirst(baseCSS + " div.sign_module div.signing_body div.videocontent link")
                .attr( attributeKey: "href").toString();
        return baseURL + videoURL;
    }
}
```

4.2.6 *Graphical User Interface (GUI)*

The users of Sign-IT can use this paper model via a graphical user interface. This section demonstrates a mock of the GUI and describes its main features.

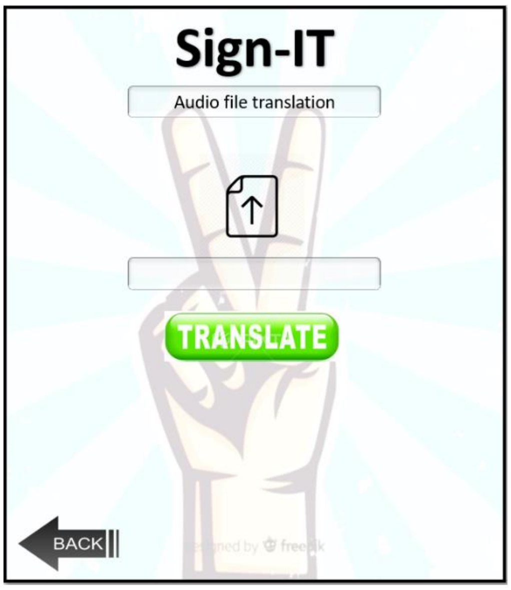
<u>Main Screen:</u> In this screen the user can choose whether to translate an existing audio file or use the microphone for a real time translation.

By using the exit button, the user exits the software.



Mock of Main Screen

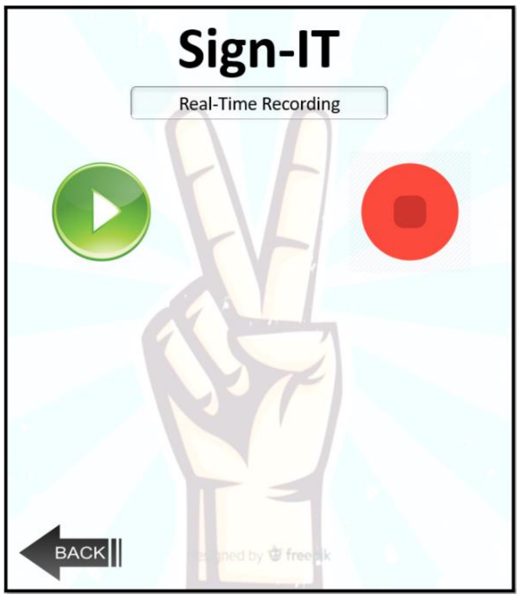
<u>Existing audio file translation screen:</u> Pressing the 'Translate Existing Audio File' button on the main screen leads to this screen. In this screen the user gets an ASL translation for an existing audio file, in order to translate an existing audio file, the file must end with mp3, wav formats, otherwise the user will get an error message. After the user uploaded a legit file, by pressing the translate button, Sign-IT will display the translation from English to ASL by sequence of videos. In case there is an ambiguity for a word that may change the meaning of the sentence, the software will demand for clarification by showing a screen with various options and context that the user have to choose one of the options in order to get the translation. After the translation completed the software will return to the Main screen.By using the 'Back' button the user returns to the Main Screen.



Mock of Existing Audio File Translation Screen

<u>Real-Time recording translation screen:</u> Pressing the 'Translate Real-Time Recording' button on the main screen leads to this screen. In this screen the user gets an ASL translation for a real-time recording by using the users' microphone. In order to translate a real-time recording, the user presses the 'Start Recording' button, then start to record the message. While the user presses the 'Stop Recording' button Sign-IT will display the translation from English to ASL. In case there is an ambiguity for a word that may change the meaning of the sentence, the software will demand for clarification by showing a screen with various options and context that the user have to choose one of the options in order to get the translation. After the translation completed the software will return to the Main screen.

By using the 'Back' button the user returns to the Main Screen.



Mock of Real-Time Recording Translation Screen

5. Evaluation/Verification Plan

System Testing (ST) is a black box testing technique performed to evaluate the complete system, the system's compliance against specified requirements. In System Testing, the functionalities of the system are tested from an end-to-end perspective.

This project uses this testing approach in order to test the correctness desired operation of the software presented in section 4. The use of this approach ensures that the software does exactly what it is designed to do.

In addition, we also perform white box tests, which will be reflected in the methods and functions that expects an input, in order to check the integrity of the input.

The table below shows the tests this project includes.

Test No.	***Test Subject***	***Expected Result***
1.	*The user press 'Translate Existing Audio File' button.*	*The screen displays the Existing Audio File translation screen.*
2.	*The user press 'Translate Real-Time Recording' button.*	*The screen displays the Real-Time translation screen.*
3.	*The user press 'Exit' button.*	*The software closes.*
4.	*The user press 'Back' button.*	*The software returns to the main screen.*
5.	*The user press 'Translate' button without uploading any input file.*	*The 'Translate' button will be disabled until the user upload the input file.*
6.	*The user uploads an audio file in the wrong format.*	*The 'Translate' button will be disabled and the software alert the user to insert an audio file in the correct format.*
7.	*The user uploads an audio file larger than 100MB.*	*The 'Translate' button will be disabled and the software alert the user to insert an audio file in the correct size.*
8.	*The user uploads an audio file in a language other than English.*	*An error message appears to the user*
9.	*The user press 'Start Recording' button and doesn't record anything for 15 seconds*	*The software stop listening and alert the user.*
10.	*The user press 'Start Recording' button and recording in a language other than English.*	*An error message appears to the user*
11.	*The user press 'Stop Recording' button before recorded anything.*	*The 'Stop Recording' button will be disabled until 'Start Recording' button pressed.*
12.	*The user recording a long message more than 1 minute.*	*The software stops the recording when the user recording more than 1 minute.*
13.	*The user press 'back' button while recording.*	*While recording all the buttons will be disabled.*
14.	*The user press 'Start Recording' button more than one time.*	*While pressing 'Start Recording' button all other buttons disabled except the 'Stop Recording' button.*
15.	*The user changes the path of an audio file after uploaded it.*	*The text field will only display the inserted path without the option to change it.*

6. References

1. *SIGN LANGUAGE TRANSLATOR USING MACHINE LEARNING Vishwas S, Hemanth Gowda M, Vivek Chandra H N, Tannvi Students, Department of Computer Science & Engineering, Vidyavardhaka College of Engineering, P.B. No. 206, Kannada Sahithya Parishath Road, III Stage, Gokulum, Mysuru, Karnataka-570002*
2. *SIGN LANGUAGE CONVERTER Taner Arsan and Oğuz Ülgen Department of Computer Engineering, Kadir Has University, Istanbul, Turkey*
3. *Text to Sign Language Conversion by Using Python and Database of Images and Videos Pooja Balu Sonawane, Anita Nikalje, Assistant Professor.*
   Department of Electronics and Telecommunication Engineering Marathwada Shikshan Prasarak Mandal's Deogiri Institute of Engineering & Management Studies, Aurangabad Maharashtra, India.
4. *Real Time Translator for Sign Languages Zulfiqar A. Memon, M. Uzair Ahmed, S. Talha Hussain, Zahid Abbas Baig, Umer Aziz Department of Computer Science National University of Computer and Emerging Sciences (FAST-NUCES), Karachi*
5. *Text Annotation with OpenNLP and UIMA Graham Wilcock University of Helsinki*
6. *SimpleNLG: A realisation engine for practical applications Albert Gatt and Ehud Reiter Department of Computing Science University of Aberdeen Aberdeen AB24 3UE, UK*
7. *WordNet: a lexical database for English*
8. https://www.startasl.com/american-sign-language/
9. https://www.newsweek.com/asl-day-2019-american-sign-language-1394695
10. https://www.tutorialspoint.com/opennlp/opennlp_pdf_version.htm
11. https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python
12. https://www.lifeprint.com/asl101/pages-layout/nonlinguisticcommunication.htm