Speech recording on front end and Speech Recognition on backend

# Goal

1. To record an audio on the front end (as a Wav file).
2. Read and encode the WAV file data to send it over WebSocket.
3. Convert the received data to text using MS Speech API.

# APIs/Plugins tested

## RecordRTC

RecordRTC is a WebRTC JavaScript library for audio/video as well as screen activity recording.

## Supported Platforms

* Android – audio + video
* iOS- audio + video
* Windows – only audio
* Browser- audio + video

### Pros

* Lot of documentation
* Easy to use

### Cons

* Even though the documentation says that RecordRTC supports safari (which is true), it does not support iOS WebView. Hence this API cannot be used.

## Cordova Plugin Media Capture

This plugin provides access to the device's audio, image, and video capture capabilities.

### Supported Platforms

* Android
* iOS
* Windows
* Browser

Pros

* Records the file and gives back the location in which the file is stored.
* Works perfectly with iOS WebView.

### Cons

* Documentation is sufficient but doesn’t clarify usage of few methods.
* We do not need to play the audio on frontend, but in case if that is also a requirement, then the code to access the file and play it must be written manually (unlike cordova plugin media).
* Multiple buttons to click to record an audio. Also a default user interface pops up upon calling the plugin.

## Cordova Plugin Media

This plugin provides the ability to record and play back audio files on a device. To read the stored file we use **Cordova Plugin Files**.

### Supported Platforms

* Android
* iOS
* Windows
* Browser

### Pros

* It’s a Cordova plugin, so they work well on mobiles/tablets.
* Easy to use
* Enough documentation

### Cons

* While the recording might be easy using this plugin, accessing the recorded file to send it to backend is difficult. Instead of using the default file location to record an audio (the location in which the file will be stored might be difficult to read), it is advised that the programmer set the location in which the file will be stored.

## Decision on API/Plugin

We need to record the audio and send it to the backend. This means we will have to access the recorded file which is stored locally. Keeping in mind these factors, we chose cordova plugin media. In this prototype, the recorded files have been stored in ‘cdvfile://localhost/temporary’.

# Steps

## Create UI

The user interface is controlled by angular. We add a start recording button to trigger the audio recording. Since we are using cordova plugin media for audio recording, the button click triggers the audio recording through media plugin.

## Using Cordova plugins in Angular code

Cordova plugins are not equivalent to npm packages which can be used with angular. They are platform specific i.e., one cannot use Cordova plugins in Angular directly. In order to use the cordova plugin, one must treat the plugin as a normal JavaScript file. The following steps describe how it can be done:

1. Create a external JavaScript file which contains functions which use cordova plugins methods.

For Example :





function StartRec()

{

mediaElement.startRecording();

//startRecording is a cordova plugins method

}

1. Add the JavaScript file in assets folder of angular project.
2. In angular.json file, add the JavaScript file as shown 
3. In the .ts file declare all the functions of JavaScript file as constants and then use them as shown below.





declare const StartRec: any; // name of constant should be same as functions name // in JS file

initRecording()

{

StartRec();

}

## Encoding Audio and sending it through a WebSocket

The recorded audio must be sent to backend for processing, which is in C#. WebSocket can be used to send string, binary or image data to the backend. Since we cannot send audio file to the backend directly, the audio file is encoded and sent. The following block diagram explains the transmission.







To encode the audio we use filereader or btoa. The usage of both has been given below

var reader = new FileReader(); 

reader.onloadend = function(encodeFile){

var src= encodeFile.target.result;

src= src.split(“base64,”);

var fileInBase64Format = src[1]; //encoded string is stored in fileInBase64Format variable





fileInBase64Format = btoa(unescaped(encodeURIComponent(file)));

//encoded string is stored in fileInBase64Format variable

## Decoding the base64 encoded string and converting it back to wav file

The encoded string must be decoded and converted back to a wav file or wav stream to pass it to MS Speech library. MS Speech library does not support other audio file formats. This can be done as follows:





var x= encodedString;

byte[] a= Convert.FromBase64String(x);

File.WriteAllBytes(filepath,a)

The length of a base64 encoded string is always a multiple of 4. If it is not a multiple of 4, then ‘=’ characters should be appended until it is. The following code isn’t necessary. But if we get any errors regarding size of the string, it can be used to.



int mod4 = x.Length % 4;

if (mod4 > 0)

{

x += new string('=', 4 - mod4);

}

## Speech recognition using MS Speech API

The audio file now should be sent to MS Speech Api to convert it from speech to text. MS Speech Api refers to the grammar ( which is the set of words which are to be recognized). The entire process can be summarized as follows







## Continuous speech recognition

The application should recognize the speech continuously. This means that the application should record speech continuously and send it to backend at regular intervals of time. The following has been done to handle this condition:

### On Frontend

The recording function has been put on a loop. So until the user chooses to disable the speech recording feature, the recording will be going on will be sent to back end at regular intervals of time.

### On Backend

In backend, every time the socket receives the message from front end, it sends the message to speech recognition.

## Running the application

Since we are using cordova plugins, this application cannot run on angular. Instead the following must be done.













For WebSocket to work, make sure that both the frontend and backend are on the same network.

Run the C# code.

## Privacy permissions on iOS

Since we will be using microphone of iPad to record the audio, it is required to ask user permission to use microphone. The easiest way to do it is to open info.plist file present in the cordova project directory as a text file and inserting the microphone permission as follows

<key>NSMicrophoneUsageDescription</key>

<string>${PRODUCT\_NAME} Some description stating why we are using microphone</string>

#### Note 1:

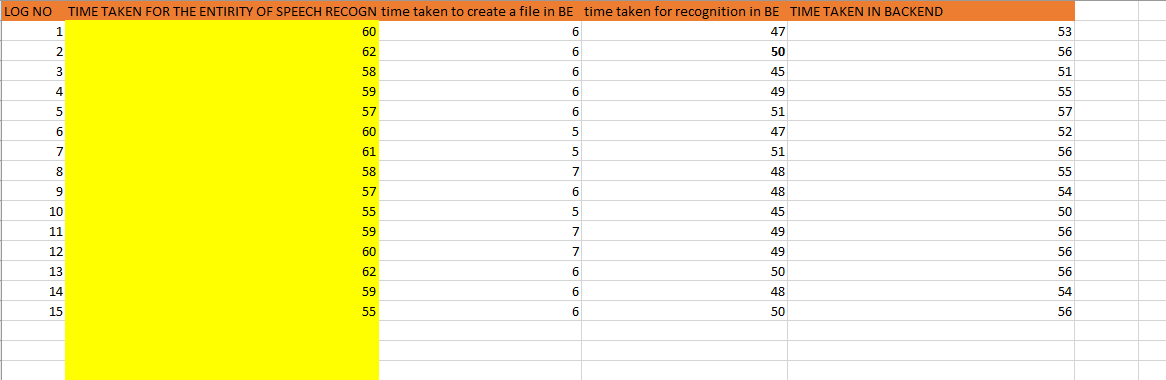
No need to specify what the product name is. The system automatically recognizes the app.

#### Note 2:

Description on why the microphone is being used is mandatory. Without that the application will be rejected.

# Performance logs

The performance logs this prototype can be found below:



*Time in milliseconds*

# Conclusion

This study explores various APIs/plugins that can be used for speech recording on frontend. Also, the use of WebSocket to communicate audio data has been demonstrated. Apart from a few privacy related settings that must be made, Speech Recording using cross platform tools like cordova works. But the plugins must be carefully chosen as not all of them perform all the tasks that a programmer might expect.