**User Identification Testing & Speech-to-Text : Report**

Rocket Elevators wants to go the “extra mile” and to launch a new features for its elevators : a “Speaker Recognition” function that will enable the users to sign in with their profile ID with their voice. If not already registered in the system with a profile ID, the user can sign up (also vocally) with a sequence of steps that will be detailed below. A second objective of Rocket Elevators is to be able to transcript conversations in its elevators to actual text.

The present report will detail the inscription and recognition sequences a user must follow to use the system. It will also describe the tests and steps that had to be realized in order for the user profiles to be saved in the system and recognized. It will then detail the procedures required to render those audio conversations in the text form with Azure Speech-to-Text.

The Microsoft Cognitive Services were used for that purpose. The system has to be taught which voice profiles to recognize as well as the vocal signatures related to it. The profiles have to be created following the cognitive services documentation (detailed on <https://westus.dev.cognitive.microsoft.com/docs/services/563309b6778daf02acc0a508/operations/5645c068e597ed22ec38f42e>).The instructions that were followed were in the “Identification Profile” and “Speaker Recognition” sections of the documentation.

The first part of the report, “Speaker identification”, will detail 10 audio conversations from the elevators and try to identify one or many speakers in those conversations. The second part of the report, “Speech-to-text” will take those 10 conversations and try to render them in a text form. The final part of the report, “Methodology”, will detail the methods and steps that were needed for the user profile creation, its enrollment, the fetching of its profile, and finally the identification and confirmation of that user. All the tests of this report have been made on the Postman application to make sure all the calls are ready to be integrated to our Ruby on Rails Web application.

**Speaker identification**

This part of the report will analyze 10 different conversations in the form of an audio file and try to identify (or not) different registered “profile IDs” (see the “Methodology” part of the present report). Each conversation will be analyzed with the following criteria :

1. Main objective
2. Choice of implementation and parameters
3. Result for every file
4. Conclusions and recommendations if the experience was to be repeated

**Conversation #1, Audio file : Conversation-1.wav : a conversation between 3 people, none of them being a user with a profile ID**

1. *Main objective* : see the result

**Methodology**

The main objectives of this project are : creating three user profiles and be able to recognize them in various contexts (mainly conversations). We also then want to be able to identify a selected user profile and be able to put conversational speeches to text with Azure Speech-to-text. The user profiles were created using the Microsoft Cognitive Services documentation methodology with the help of the Postman app allowing us to make the required API calls to create the user profiles and then verify their actual integrity (for example, are the right users correctly identified ?). All the vocal samples used for the profile creations were English or French discourses where the subjects recited song lyrics on a conversational tone for a duration of about 40 to 60 seconds. The ten conversations used for the speaker recognition part of this project were “elevator-related business conversations” in French and in English. Every test detailed below will be analyzed with the following criteria :

1. Main objective
2. Choice of implementation and parameters
3. Result for every file
4. Conclusions and recommendations if the experience was to be repeated

Every step was completed using the Microsoft Cognitive Services documentation that can be found here : <https://westus.dev.cognitive.microsoft.com/docs/services/563309b6778daf02acc0a508/operations/5645c068e597ed22ec38f42e>

**User profile creation**

1. *Main objective* : create three different user profiles, two English and one French. Raphaël Vallée et Maxime Patry are the English-speaking users. Rémi Lemay-Dupont is the French-speaking user.
2. *Choice of implementation and parameters* : we created the user profiles using the Postman app to call the following GET url with a POST function : **https://westus.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles**

The following parameters had to be entered in the Postman application :  
-in the “Headers” section, the following key/values pairings : Content-type / application/json, Accept / application/json, Ocp-Apim-Subscription-Key / a9e3e21c91124fc3bb63a9e8ec9813a4 (related to the Azure Cloud account)  
-in the “Body” section, enter the following text : { “locale” : “en-us” }, select “raw” and “JSON” as a type (the “fr-fr” parameters were entered for user 3, a french-speaking user).

1. *Result for every file* : after pressing the “Send” button, the response gave back an “identificationProfileId” in the following form : { “identificationProfileId” : “49a36324-fc4b-4387-aa06-090cfbf0064f” } which will be needed for the subsequent steps.
2. *Conclusions and recommendations* : this being the first step (and a very simple one), and the documentation being crystal clear on this, it wasn’t hard to complete.

**Enrollment creation**

1. *Main objective* : associate each created profile with a voice do the speaker can later be identified. A 30 to 45 seconds voice sample was recorded to be sent and processed in the instructions below.
2. *Choice of implementation and parameters* : we enrolled the different user profile IDs using the following url with a POST function : **https://westus.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58/enroll/**

Here, the “1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58” in the url was a profile ID created in the earlier step. The following parameters had to be entered in the Postman application :   
-in the “Headers” section, the following key/values pairings : Content-type / multipart/form-data, Ocp-Apim-Subscription-Key / a9e3e21c91124fc3bb63a9e8ec9813a4.  
-in the “Body” section, the recorded voice sample that we wanted to associate the profile ID to was sent as a binary file that we had to send along with the POST function. The voice samples were recorded using a Blue Yeti microphone as well as the Audacity software. The samples were song lyrics recited by the users on a conversational tone.

1. *Result for every file* : we ran into a few issues while transmitting the audio files. We firstly tried to send a too short (28 seconds) sample to the url, with an 404 error message returned to us telling us the sample was too short. Another try gave us another error message because the file had to be in mono. Another try returned us a 404 message saying the sample had to be in 16K format. We retried the B) steps with a 16K, mono and longer audio sample, which returned a 202 OK response. Therefore, we only ran into issues for user 1’s enrollment, because we recorded longer and appropriately formatted audio samples for the others (about 40 seconds long).
2. *Conclusions and recommendations* : audio files should be at least 35 seconds long to avoid recording it twice. The enrollment result will be verified with in the subsequent section : “Profile fetching”.

**Profile fetching**

1. *Main objective* : see if the profile creation and enrollment procedures now allow us to “GET” a profile and see its properties.
2. *Choice of implementation and parameters* : we fetched the user profiles using the following url with a GET function : **https://westus.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58**

Where the “1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58” in the url was the profile ID we wanted to fetch, to “GET”. The following key/value parameters had to be entered in the “Headers” section of the Postman request : Ocp-Apim-Subscription-Key / a9e3e21c91124fc3bb63a9e8ec9813a4. The GET request could now be sent to the url.

1. *Result for every file* : All our profiles were rightly created, so our query returned the following in the “Body” section of Postman :

{

"identificationProfileId": "1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58",

"locale": "en-us",

"enrollmentSpeechTime": 56.19,

"remainingEnrollmentSpeechTime": 0,

"createdDateTime": "2019-04-29T16:50:43.513Z",

"lastActionDateTime": "2019-04-29T18:16:34.600Z",

"enrollmentStatus": "Enrolled"

}

The results varied from user to user (because they have different “identificationProfileIds”, because they were not created at the same time and because user 3 has “fr-fr” instead of “en-us” in the “locale” section), but all returned 200 OK responses. They were all “Enrolled”, which means the enrollment did work.

1. *Conclusions and recommendations* : all profiles returned “Enrolled” the first time we tried to “GET” them, so no further recommendation will be suggested here. Very few and very simple steps have to be realized and are all crystal clear in the Microsoft Cognitive Services documentation.

**User identification**

1. *Main objective* : identify if a chosen user (with its profile ID) is present or not in a specific audio sample sent to an identification url. The Microsoft API will then analyze if the profile ID X can actually be heard in the sample and other data that we will describe to you below.
2. *Choice of implementation and parameters* : we sent the user profiles to be identified using the following url with a POST function : **https://westus.api.cognitive.microsoft.com/spid/v1.0/identify?identificationProfileIds={identificationProfileIds}**

Where the “{identificationProfileIds}” in the url was the profile ID we wanted to identify, which has to be entered as a key/value pair in the “Params” section of the Postman application. The following key/value parameters had to be entered in the Postman request :   
-in the “Params” section, identificationProfileIds/1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58 where “1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58” is the user 1’s profile ID (for the purpose of this example). It should be noted that up to 10 profiles can be identified at the same time in a single request.   
-in the “Headers” section, the following key/values pairings : Content-type / multipart/form-data, Ocp-Apim-Subscription-Key / a9e3e21c91124fc3bb63a9e8ec9813a4.  
-in the “Body” section, a new recorded voice sample that we wanted to identify the profile ID with was sent as a binary file that we had to send along with the POST function. The voice samples were recorded using a Blue Yeti microphone as well as the Audacity software. The “ID-verification” samples were different samples from those used in the “Enrollment creation” section and were improvised speeches or other song lyrics recited on a conversational tone. The samples were all 35 seconds or more.

1. *Result for every file* : When we pressed “Send”, a 202 accepted response appeared in the Postman dashboard and new headers were rendered in response :

**Cache-Control →**no-cache

**Pragma →**no-cache

**Content-Length →**0

**Expires →**-1

**Operation-Location →**https://westus.api.cognitive.microsoft.com/spid/v1.0/operations/5dee28ac-bf86-4b7c-9584-cd99ef201f31

**X-AspNet-Version →**4.0.30319

**X-Powered-By →**ASP.NET

**apim-request-id →**3dadcfed-d12c-406c-b466-554a596e0273

**Strict-Transport-Security →**max-age=31536000; includeSubDomains; preload

**x-content-type-options →**nosniff

**Date →**Wed, 01 May 2019 19:49:44 GMT

Here, the only thing that we need and that varies from user to user is the “Operation-Location” url which will then be used in the subsequent “User confirmation” step to check the result of the user identification.

1. *Conclusions and recommendations* : As the Microsoft Cognitive Services documentation puts it, each recorded speech that we want to verify should be at least 30 seconds. The conclusion of this step will then be clearer in the subsequent step where we get to check the status of our identification.

**User confirmation**

1. *Main objective* : Check the status of the precedent step’s request to see if the user(s) has rightly been identified.
2. *Choice of implementation and parameters* : we sent the “operationId” used to identify a user using the following url with a GET function :

<https://westus.api.cognitive.microsoft.com/spid/v1.0/operations/5dee28ac-bf86-4b7c-9584-cd99ef201f31>

Where the whole url that we want to GET is the result of the”Operation-Location” header rendered in the precedent step. The only key/value pairing that needs to be entered here is the Ocp-Apim-Subscription-Key / a9e3e21c91124fc3bb63a9e8ec9813a4. An “operationId” key/value pairing then appears by default in the “Params” section of the Postman request.

1. *Result for every file* : When the “Send” button was pressed, the response returned in the “Body” part of every request was similar to the following :

{

"status": "succeeded",

"createdDateTime": "2019-04-29T18:58:21.4717298Z",

"lastActionDateTime": "2019-04-29T18:58:23.627657Z",

"processingResult": {

"identifiedProfileId": "1bb58f2d-7db2-4b5d-8d58-7a5ea7aece58",

"confidence": "High"

}

}

Where only the time stamps and “identifiedProfileId” changed from user to user. The confidence level for every file was “High”. We tried sending a sample where another user was speaking, but we asked for a different user to be identified. This test gave the following result :

{

"status": "succeeded",

"createdDateTime": "2019-04-29T19:08:21.4717298Z",

"lastActionDateTime": "2019-04-29T19:08:23.627657Z",

"processingResult": {

"identifiedProfileId": "00000000-0000-0000-0000-000000000000",

"confidence": "Normal"

}

}

The Microsoft Cognitive Services documentation is clear on this kind of result (<https://westus.dev.cognitive.microsoft.com/docs/services/563309b6778daf02acc0a508/operations/5645c725ca73070ee8845bd6>) : the API cannot recognize the provided profile in the provided audio sample.

1. *Conclusions and recommendations* : This step being the final one of the “Speaker Recognition” part of our project, a few conclusions and recommendations can be provided :   
   -CLOSELY read the Microsoft Cognitive Services documentation and follow all the steps and you should have no problem. Every issue that we ran into was us going too fast and passing over clear instructions like “samples should be AT LEAST 30 seconds long, in mono and in the 16K format”.   
   -All the expected responses and scenarios are provided in the documentation. You are having a 404 response or an unsatisfying response in the body of a 202 response ? Chances are the exact result can be found in the documentation to tell you what may have been wrong with your request.

-There can be a delay to “GET” the identified user confirmation. Simply sending the operation ID url another time did the trick.

-Many operations are possible with the Speaker Recognition API, make sure you are using the right url to get the wanted request (for example, do not mix-up “Identification Profile” and “Verification Profile”, which are two completely different methods).