Sign Language Translation Mobile Application and Open Communications Framework

Deliverable 2.2: SignON Services Framework Architecture

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101017255

Project Information
Project Number: 101017255
Project Title: SignON: Sign Language Translation Mobile Application and Open Communications Framework
Funding Scheme: H2020 ICT-57-2020
Project Start Date: January 1st 2021

Deliverable Information
Title: D2.2 - SignON Services Framework Architecture
Work Package: WP2 - SignON service and mobile app
Lead beneficiary: FINCONS
Due Date: 31/08/2021
Revision Number: V1.0
Authors: M. Giovanelli (FINCONS), M. P. Scipioni (FINCONS)
Dissemination Level: Public
Deliverable Type: Other

Overview: The aim of this document is to define the SignON Framework architecture and specify an appropriate repository for the inference data (temporary storage for user input processing) and correction data (permanent storage for the retraining of models).

Revision History

Version #	Implemented by	Revision Date	Description of changes
V0.1	FINCONS	09/07/2021	First draft
V0.2	FINCONS	27/07/2021	Add contributions from MAC and Requirements chapter
V1.0	FINCONS	27/08/2021	Final version

The SignON project has received funding from the European Union's Horizon 2020 Programme under Grant Agreement No. 101017255. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SignON project or the European Commission. The European Commission is not liable for any use that may be made of the information contained therein.

Approval Procedure

Version #	Deliverable Name	Approved by	Institution	Approval Date
V1.0	D2.2	Aoife Brady	DCU	23/08/2021
V1.0	D2.2	Bob Boelhouwer	INT	05/08/2021
V1.0	D2.2	Adrian Nuñez Gorka Labaka	UPV/EHU	11/08/2021
V1.0	D2.2	John O'Flaherty	MAC	28/07/2021
V1.0	D2.2	Euan McGill	UPF	16/08/2021
V1.0	D2.2	Irene Murtagh	TU Dublin	27/08/2021
V1.0	D2.2	Lorraine Leeson	TCD	11/08/2021
V1.0	D2.2	Gregg Young	VRT	23/08/2021
V1.0	D2.2	Mathieu De Coster	UGent	16/08/2021
V1.0	D2.2	Jorn Rijckaert	VGTC	24/08/2021
V1.0	D2.2	Anthony Ventresque	NUID UCD	30/08/2021
V1.0	D2.2	Henk van den Heuvel	RU	19/08/2021
V1.0	D2.2	Catia Cucchiarini	TaalUnie (NTU)	23/08/2021
V1.0	D2.2	Tim Van de Cruys	KU Leuven	24/08/2021
V1.0	D2.2	Frankie Picron	EUD	23/08/2021

V1.0	D2.2	Mirella De Sisto Dimitar Shterionov	TiU	16/08/2021 17/08/2021

Acronyms

The following table provides definitions for acronyms and terms relevant to this document.

Acronym	Definition
API	Application Programme Interface
ASR	Automatic Speech Recognition
BML	Behaviour Markup Language
DHH	Deaf and Hard of Hearing
JSON	JavaScript Object Notation
MT	Machine Translation
NLU	Natural Language Understanding
POS	Part of Speech
REST	Representational State Transfer
SL	Sign Language
SOTA	State of the Art
TTS	Text-To-Speech
URL	Universal Resource Locator

Table of Contents

Introduction **7**

Requirements **8**

SignON App Requirements 8

SignON Pipeline Technical Requirements 11

Input requirements for the SignON Pipeline 11

Sign Language Recognition Requirements 12

Automatic Speech Recognition Requirements 12

Natural Language Understanding Requirements 13

Output Requirements for the SignON Pipeline 14

Architecture **16**

Overview 16

Interfaces 18

SignON App - SignON Orchestrator 18

Standard Interaction 18

Correction Interaction 20

SignON Orchestrator - SignON Pipeline 21

SignON Orchestrator 23

Standard Interaction 24

Correction Interaction 25

SignON Pipeline 26

Source Language Processing 27

Intermediate Representation 29

Message Synthesis 31

Conclusion **33**

1. Introduction

The SignON project will research and develop a communication service that uses Machine Translation (MT) to translate between sign and spoken languages. This service will facilitate the exchange of information between Deaf and Hard of Hearing (DHH), and hearing individuals.
To achieve this result, the SignON project will develop a free, open SignON App and SignON Framework for conversion between video (capturing and understanding sign language), audio and text, as well as translation between signed and spoken languages.

The aim of this document is to propose a design of the SignON Framework architecture, that will be possibly adapted to any new needs arising during the project. Furthermore, in this document an appropriate repository for the inference data (temporary storage for user input processing) and correction data (permanent storage for the retraining of models) are specified.

The SignOn services will be offered through the following components: the SignON App, i.e. the mobile app working as a frontend for users with a lightweight interface, and the SignON Framework, i.e. the cloud-based platform where the computationally intensive services will be executed. The SignON Framework will comprise a SignON Orchestrator, which will be responsible for the communication from and to the SignON App, and the SignON Pipeline, where all the processing will be performed.

Firstly, the requirements from the SignON App and the SignON Framework are listed in order to define the working background. Secondly, the architecture is presented, starting at an overview of the entire system and proceeding with details about the interfaces and the description of the internal components. Finally, a brief conclusion is provided, in order to summarise the main points of this deliverable.

For the sake of brevity, the deep technical details are left to subsequent deliverables that will describe how the implementation of all the components of the system will be realized.

2. Requirements

The current deliverable builds on and extends work carried out in WP1 in Task 1.4, "Technical User Requirements", which is described in Deliverable D1.4, "First Technical Requirements and User Research (UX design) Report". In particular, technical requirements related to the SignON App, which acts as the frontend, are summarised in D1.4. These technical requirements have been specified after the analysis of the user requirements described in D1.3, "First user Requirements Report".

Besides the technical requirements gathered regarding the frontend and users' interaction with the system (SignON App), this deliverable also focuses on requirements originating from the processing pipeline (SignON Pipeline), which acts as the backend to the system, implementing the functionalities that allow users to benefit from the SignON service.

2.1. SignON App Requirements

In this deliverable we develop technical requirements targeting the use-cases which are described in D1.1 and have been verified through the works described in D1.3.

Technical requirements for the SignON App have been gathered in Task 1.4 and in the related Deliverable D1.4, "First Technical Requirements & User Research (UX design) Report":

A. *User's Mobile Device:*
    1. *The SignON App must be easy and intuitive to use. Simple but powerful. To run on standard modern phones and tablets.*
    2. *The SignON App will be free and easily downloadable by users from the Google Play Store for Android phones and tablets, and from the Apple App Store for iPhones and iPads.*
    3. *All the SignON App's user's inputs and outputs will be on a single mobile user device to communicate in-person with people nearby.*
    4. *Future versions may interwork with a messaging app (such as WhatsApp) on the same user device, to remote people.*
    5. *The User's Mobile Device must have broadband data Internet access for the App to operate using the SignON Framework cloud-based services.*
        a. *Future versions may provide limited off-line operation, such as a user-selectable vocabulary of Signing GIFs, if requested by users.*

B. *System Performance*

1. *Translation/conversion will be unidirectional operation with users taking turns to input their messages of up to 30 seconds duration.*
    a. *Users may choose to store messages on their own device, but messages will not be retained by the Framework cloud system. Data privacy and protection will be explicitly stated to the user.*
    b. *Video, audio, or text streaming will not be provided.*
2. *Translation/conversions and user login/authorisations should:*
    a. *Respond within 2 seconds – with a maximum of 5 seconds for SL-to-SL translations, to enable effective user communications.*
    b. *Provide user-acceptable accuracy for 75% of users.*
3. *Users should have at least 75% average satisfaction rating with the overall operation of the SignON service.*
4. *The SignON App will automatically and securely login and authorise the user to the SignON Framework services, after a one-time short and simple manual user setup and authorisation.*

C. *User Preferences*

1. *The SignON App will provide one-tap user-selectable translation and conversion between any combination of:*
    a. *Flemish Sign Language (VGT), Sign Language of the Netherlands (NGT), Irish Sign Language (ISL), British Sign Language (BSL) and Spanish Sign Language (LSE).*
    b. *English, Irish, Dutch and Spanish spoken and text languages.*
2. *The App will provide, and retain, user selectable*
    a. *default personalised options for the User Interface, App text languages and favourite settings.*
    b. *UI display, audio and text options, including contrast and SL avatar customisation.*

D. *Sign Language Translation*

1. *User SL Input*
    a. *No additional special attachments or special lighting will be required for capturing SL input. The App will automatically adapt to input ambient light conditions, within the limits of the user's device.*

b. *User-selectable use of either selfie or forward-facing device video camera live, or a pre-recorded video.*
c. *SLR accuracy and operation to be acceptable to 75% of users:*
    i. *Accommodating both formal and informal styles.*
    ii. *Covering regional signs, age variation & fingerspelling.*
    iii. *Recognising emotion (through facial expression and signing style).*
    iv. *Recognising and translating classifiers and using an appropriate lexicon.*
    v. *Attending to specific SL grammar.*
d. *Future versions may include a user option to add new signs, or have SignON learn them through repeated use of certain signs.*
e. *Performance better than the SignAll automatic translation of ASL to English for all of SignON's supported SLs.*

2. *User SL output*
    a. *A user acceptable and customisable 3D virtual signer, which focuses on linguistic accuracy. This means accurate hand forms, hand and finger movements, body movement, posture, the right speed and facial features such as showing the right emotion, lip movement, eyebrows and eye gaze. Apart from this, the avatar must be user customisable concerning gender, skin colour and contrasting colours in clothing and background.*
    b. *Have a user option to include message text to confirm and correct the accuracy of the avatar's signing.*
    c. *Overall performance better than the SiMAX avatars.*

E. *Speech and Text Translation*
1. *For normal and atypical, formal and informal, speech.*
2. *User preselection, and an option for automatic detection of the user's input text and speech languages.*
3. *Indicate visually that speech is being recorded and played back (for DHH users).*
4. *User option to store conversation message texts on the user's device.*
5. *Future versions may translate/answer phone calls, interpret emotions and ambient background noise, and use symbols or simple text.*
6. *Overall performance is better than Google Translate.*

2.2. SignON Pipeline Technical Requirements

Besides requirements originating from the design of the SignON App, technical requirements originating from the SignON Pipeline are also considered.

The SignON Pipeline is comprised of three main components:

- Source Language Processing (WP3),
- Intermediate Representation (WP4),
- Message Synthesis (WP5).

In this document, we will focus on input output requirements to and from the SignON Pipeline, i.e. input requirements for the Source Language Processing component and output requirements from the Message Synthesis component. Requirements related to the internal representation of the SignON Pipeline will be dealt with in detail in future deliverables and will not be taken into account in this document.

For each of these components the main requirements are highlighted in the following subsections.

2.2.1. Input requirements for the SignON Pipeline

The SignON Pipeline will receive data originated from the SignON App for each new translation request. Data originated from the SignON App can be transferred according to three different modes of communication

- Sign language,
- Audio,
- Text.

Each mode has different technical requirements and will be processed by different sub-components of the SignON Pipeline. In this section, requirements for each sub-component, and thus related to different modes, are analysed separately.

2.2.1.1. Sign Language Recognition Requirements

The first sub-component deals with sign language input to recognise its content and process it further within the pipeline. As input, this sub-component expects a video file, such as an MP4 file, with corresponding structured metadata. Metadata will be specified in JSON format, and will contain information such as an identifier linking the metadata to the input video, the source language ID, the signer ID, the target language ID and the target mode (sign language, audio or text).

The sub-component will provide the processed information in JSON format, containing metadata and SLR output, which will be processed by other components of the pipeline. The exact format is subject to change based on research findings, but it will contain information such as hand shape, movement, eye gaze, eyebrow shape, mouthing/mouth gestures, gloss and gestural location.

2.2.1.2. Automatic Speech Recognition Requirements

When input data is audio, the Automatic Speech Recognition (ASR) sub-component will convert received audio (spoken messages) into its textual form in the source language using SOTA ASR models which will be tuned to the use cases at hand and to the speech articulated by the speaker (including atypical from deaf speakers and those with cochlear implants). The resulting text output will be seamlessly fed into the Natural Language Understanding (NLU) pipelines, described in the next subsection.

The ASR sub-component will:

- address privacy challenges,
- adapt to conversational properties (e.g. gender of the user, irregular speech, domain specifics), and
- be extensible to new languages and data.

The audio input can be mono or stereo, with additional channel information. If multiple speakers are present in one audio stream, segmentation between speakers needs to be supplied as additional information to the ASR sub-component. Additional information will be supplied as metadata in a structured text format such as JSON. The metadata will contain information such as an identifier to link the metadata to the audio, segmentation between speakers (optional), the source language ID, the speaker ID, the target language ID, the target mode (sign language, audio or text).

2.2.1.3. Natural Language Understanding Requirements

When input is text, the NLU sub-component of the pipeline will process the input data to conduct several operations such as [Inverse-] text normalisation, adaptation (simplification, generation) and spelling correction, sentence and word identification and classification, Part of Speech (POS) tagging and lemmatisation, Named Entity Recognition, coreference resolution and [named] entity linking, handling of figurative expressions and parsing in the source language and discourse modeling (which may also include handling the productive lexicon unique to sign language). The development of such language specific pipelines aims to improve the understanding of the language and prepare the recognised source for inclusion in the InterL (WP4), especially in cases of under-resourced languages where case-specific design and development (using classical rule-based or statistical methods) is preferable to data-demanding deep learning approaches.

Additional information will be supplied as metadata in a structured text format such as JSON. The metadata will contain information such as the source language ID, the target language ID and the target mode (sign language, audio or text).

2.2.2. Output Requirements for the SignON Pipeline

The output of the SignON Pipeline will be provided by the Message Synthesis component (WP5). This component deals with the real-time target message synthesis in sign and spoken languages and generates output from the SignON Pipeline to the SignON App. A personalisable 3D virtual signer will be developed to convey the translated message in the target sign language, which will be codesigned with and verified by the users (WP1). This component will also encapsulate the work for written language synthesis and normalisation based on the representation in InterL (WP4). Thereafter this text will serve as the basis for synthesis of spoken language through a commercial text-to-speech platform.

It will be possible to synthesise the output of this component for the final user in the target language of choice, in one of the following three modes:

- Sign language,
- Audio,
- Text.

When sign language is chosen, the output will be a 3D visualisation using WebGL of the 3D character developed within the component. The output will be made available to the SignON App by means of a URL pointing to the 3D scene representing the translated source message.

When text is chosen, the output will be translated text normalised and transformed into a suitable output format. The output format will be JSON, so that it can easily be integrated within the SignON App.

Finally, if the output mode of choice is audio, the pipeline will still provide text as output, delegating speech synthesis to external components which can be invoked by the SignON App and their result presented directly to the user.

Additional information will be supplied as metadata in a structured text format such as JSON. The metadata will contain information such as the source language ID, the target language ID and the target mode (sign language, audio or text).

The architecture presented in the next section considers the technical requirements originating from the SignON App and from the SignON Pipeline and connects the frontend and the backend by means of an orchestrator (SignON Orchestrator), which has the goal of allowing communication between the final users and the components which are implementing the SignON Framework services.

3. Architecture

This section describes the overall architecture of the SignON project, its developed components, the message flow between those components and their mutual interactions.

3.1. Overview

As previously mentioned, the SignON project will research and develop a communication service that uses MT to translate between sign and spoken languages.

To achieve this result, the SignON project will involve a SignON App and a SignON Framework for conversion between video (capturing and understanding sign language), audio and text and multi-language translation between signed and spoken languages.

The following high-level representation (Figure 1) shows a preliminary version of the architecture that was provided in the project proposal.

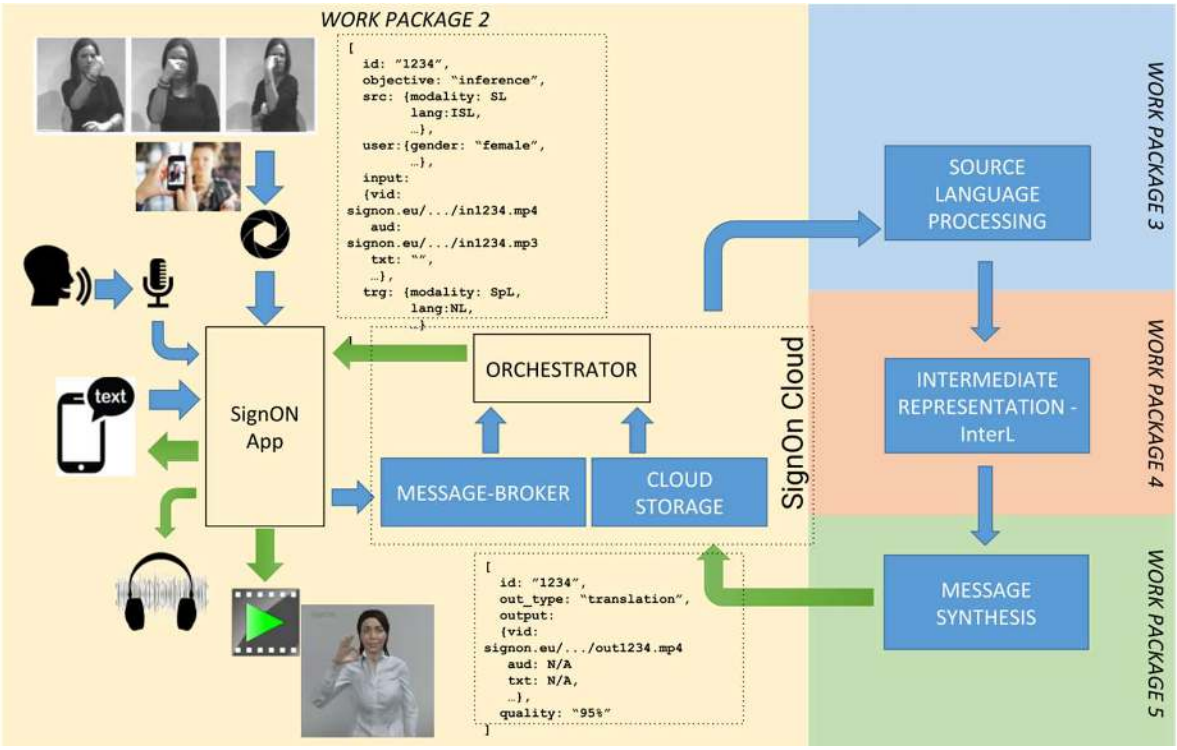

Figure 1 – SignON architecture presented in the project proposal; on the left, the SignON App and the foreseen interaction with the user; in the center, the SignON Orchestrator and its sub-components; on the right, the SignON Pipeline with its three sub-components.

From the proposed architecture, three main components can be identified (Figure 2): the SignON App, the SignON Orchestrator and the SignON Pipeline (with the latter two comprising the SignON Framework).

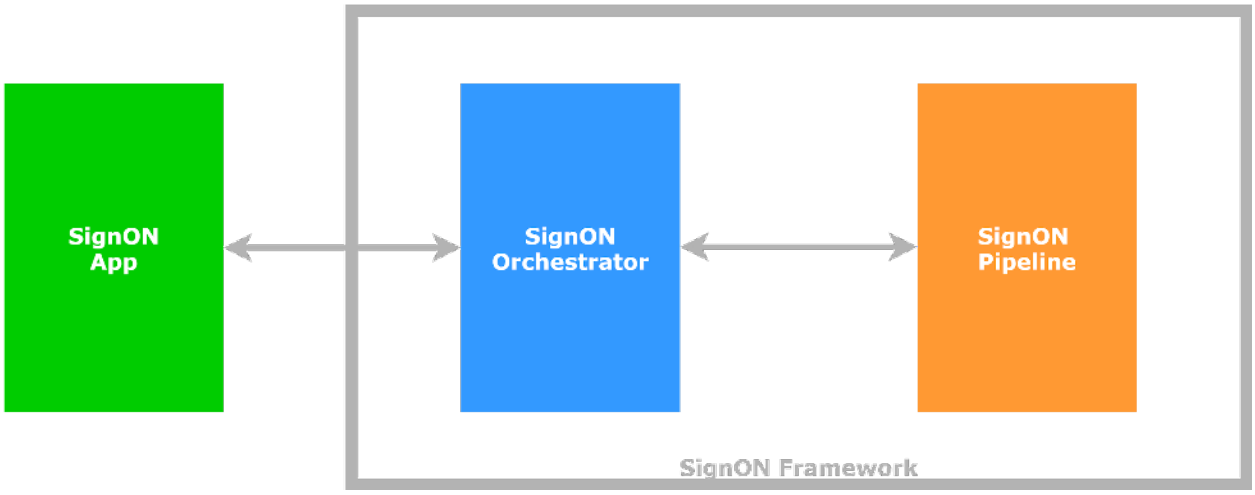

Figure 2 – SignON architecture overview.

The **SignON App** will be responsible for sending the user's input (video, audio or text) to the SignON Orchestrator and for receiving the processed output and providing it to the user in the desired format (avatar, audio or text).

The **SignON Orchestrator** will be responsible for the interactions between the SignON App and the SignON Pipeline and will manage the user authentication, the sorting and management of the messages and the storage of the data.

The **SignON Pipeline** will be responsible for the processing; in particular interpreting the input, translating it, as well as synthesising the output.

A description of the SignON Orchestrator and the SignON Pipeline can be found in the following sections, while further details about the SignON App will be released later in the project and will be presented in the following deliverables:

- *D2.6 - First release of the SignON Communication Mobile Application, due by M18 (June 2022)*
- *D2.7 - Final release of the SignON Communication Mobile Application, due by M30 (June 2023)*

3.2. Interfaces

This section describes the interfaces between the SignON App, the SignON Orchestrator and the SignON Pipeline.

3.2.1. SignON App - SignON Orchestrator

The interface between the SignON App and the SignON Orchestrator will be based on a RESTful API allowing a stateless and synchronous approach. Furthermore, for the future, this solution will enable systems other than the SignON App to be interfaced with the SignON Framework (e.g. Broadcasters' Pipelines, Messaging Applications, etc.) using the SignON API.

The two main types of interactions will be the "standard interaction" and the "correction interaction".

3.2.1.1. Standard Interaction

In the "standard interaction" the users will provide the input (video, audio or text) to the SignON App, which will send it along with metadata (e.g. user preferences, target language, target output, etc.). After the processing, the SignON App will receive the output and metadata and the SignON App will provide the desired output (avatar, audio or text) to the user. This will be the main usage of the SignON App and Framework services.

It is important to note that:

- When the requested output is audio, the SignON App will still receive text (the audio transcription) and will provide audio to the user through the usage of a TTS system.
- When the requested output is an avatar, for each translation, the SignON App will receive a specific URL to a 3D avatar scene and will display it to the user through the usage of WebGL technology.

Finally, the SignON App will locally store the input provided by the user, so the user will always be able to process a previous input again by selecting it from the history.

In the following table (Table 1), a preliminary and non-exhaustive list of the information exchanged between the SignON App and the SignON Orchestrator during the "standard interaction" is shown. For further details about the input and output requirements, please refer to the related chapter.

Table 1 – Information exchanged between the SignON App and the SignON Orchestrator during "standard interaction".

Direction	Exchanged Information
Input (from SignON App to SignON Orchestrator)	Data: ● video file / audio file / text
	Metadata: ● source language ● source mode (video / audio / text) ● source information (e.g. format, etc.) ● target language ● target mode (avatar / audio / text)
Output (from SignON Orchestrator to SignON App)	Data: ● text / audio transcription / avatar URL
	Metadata: ● transaction ID ● source language ● source mode (video / audio / text) ● source information (e.g. format, etc.) ● target language ● target mode (avatar / audio / text) ● target information (e.g. format, etc.) ● SignON Pipeline components versions

3.2.1.2. Correction Interaction

If needed, the users will be able to perform a "correction interaction" immediately after each "standard interaction" only. In this case, once users receive the system's output and they feel that it is not correct, they will be able to provide their correction (video, audio or text) to the SignON App, and the SignON App will send the provided correction along with the original input and metadata to the SignON Orchestrator. The SignON Orchestrator will then store the received data in permanent storage. For this reason, the "correction interaction" will be possible only for registered users that approved the consent to manage their collected data.

In the following table (Table 2), a preliminary and non-exhaustive list of the information exchanged between the SignON App and the SignON Orchestrator during the "correction interaction" is shown.

Table 2 – Information exchanged between SignON App and SignON Orchestrator during "correction interaction".

Direction	Exchanged Information
Input (from SignON App to SignON Orchestrator)	Data: ● input: video file / audio file / text ● correction: video file / audio file / text
	Metadata: ● user ID ● transaction ID ● source language ● source mode (video / audio / text) ● source information (e.g. format, etc.) ● target language ● target mode (avatar / audio / text) ● target information (e.g. format, etc.) ● SignON Pipeline components versions ● correction language ● correction mode (video / audio / text) ● correction information (e.g. format, etc.)

3.2.2. SignON Orchestrator - SignON Pipeline

The interface between the SignON Orchestrator and the SignON Pipeline will be based on an asynchronous approach to allow scalability and loose coupling.

The SignON Orchestrator will send to the SignON Pipeline the text input and metadata received from the SignON App. For what concerns the video and audio input, the SignON Orchestrator will provide the SignON Pipeline the temporary location on a shared storage server of the files to be processed.

After the processing, the SignON Pipeline will send back the result to the SignON Orchestrator that will forward it to the SignON App.

In the next page table (Table 3), a preliminary and non-exhaustive list of the information exchanged between the SignON Orchestrator and the SignON Pipeline is shown.
For further details about the input and output requirements, please refer to the related chapter.

Table 3 – Information exchanged between SignON Orchestrator and SignON Pipeline

Direction	Exchanged Information
Input (from SignON Orchestrator to SignON Pipeline)	Data: ● video location / audio location / text
	Metadata: ● transaction ID ● source language ● source mode (video / audio / text) ● source information (e.g. format, etc.) ● target language ● target mode (avatar / audio / text)
Output (from SignON Pipeline to SignON Orchestrator)	Data: ● text / avatar URL
	Metadata: ● transaction ID ● source language ● source mode (video / audio / text) ● source information (e.g. format, etc.) ● target language ● target mode (avatar / audio / text) ● target information (e.g. format, etc.) ● SignON Pipeline components versions

3.3. SignON Orchestrator

The SignON Orchestrator will be responsible for all SignON API interactions between the SignON App and the SignON Pipeline and will manage the user authentication, the sorting of the messages and the storage of the data. In order to accomplish all the foreseen functionalities, the SignON Orchestrator will be composed of multiple sub-components: User Authentication, API Manager, Queue, Permanent Storage and Temporary Storage.

In the following figure (Figure 3) the internal architecture of the SignON Orchestrator is shown.

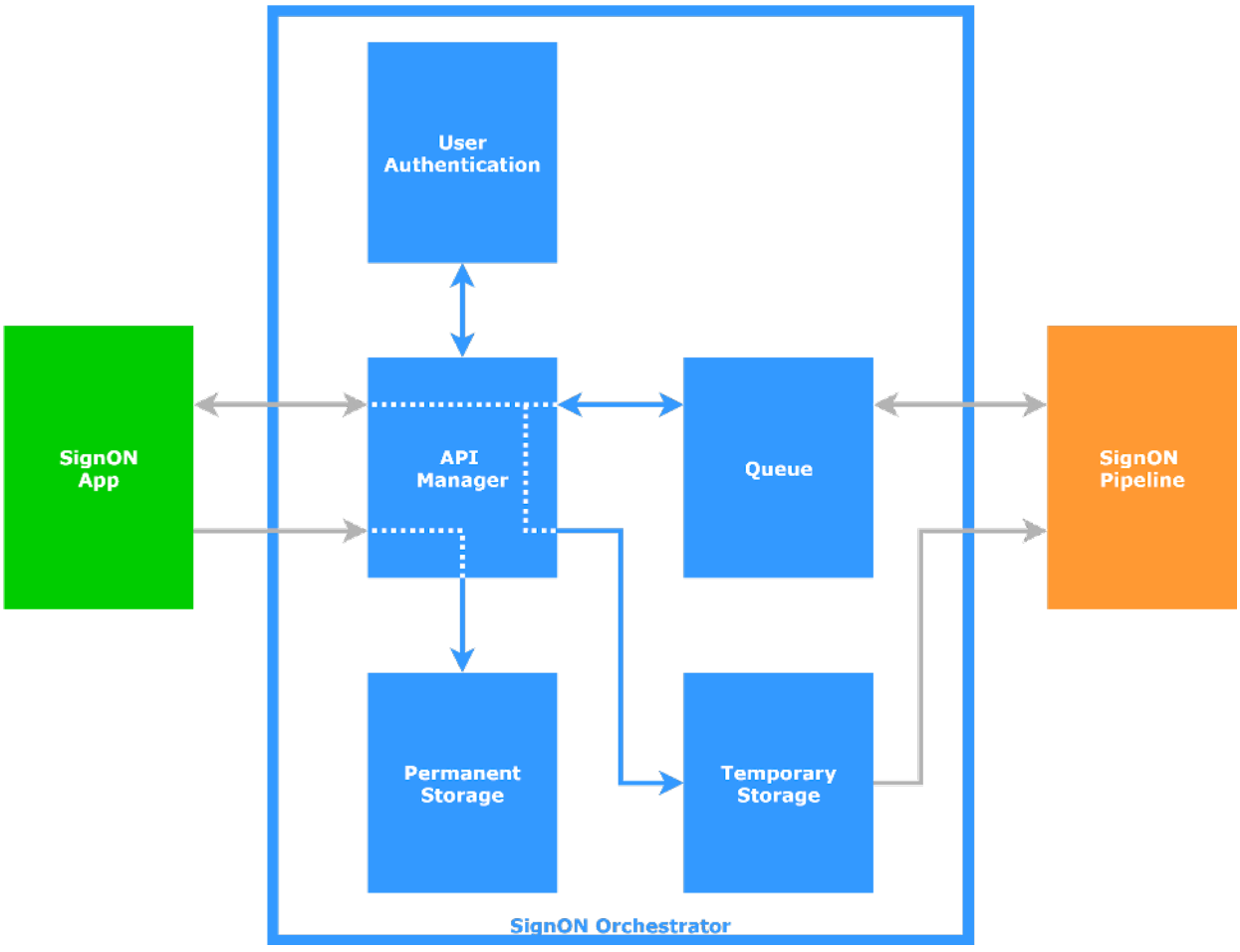

Figure 3 – SignON Orchestrator architecture overview.

As previously mentioned, two types of interactions are foreseen: the “standard interaction” and the “correction interaction”.

3.3.1. Standard Interaction

In the "standard interaction" the user will provide the input to the SignON App, which will be sent for processing and then retrieved to show each user's desired output.

To achieve this result, the following steps are foreseen:

1. The SignON App will send the user's input (video, audio or text) to the API Manager. This will be done through a RESTful API in order to have a synchronous interaction and avoid dealing with different sessions, allowing anonymous usage like the most common translation services (e.g. Google Translate).
2. Only in the case of video or audio files, the API Manager will store them in the Temporary Storage, obtaining their locations. This will be feasible through the usage of technologies such as MinIO[1], that will provide a flexible object storage (that can be initially deployed on a private solution, and eventually migrated to a cloud service provider) with a retention system to guarantee the deletion of the user's data.
3. The API Manager will prepare the message for the SignON Pipeline and will place it in the Queue, allowing an asynchronous approach to allow scalability and loose coupling. This will be accomplished through the usage of messaging technologies such as RabbitMQ[2], creating a queue for the SignON Pipeline (where the API Manager will send the message) and a temporary queue for each request of the API Manager (where the SignON Pipeline will send back the result message from the processing).
4. The SignON Pipeline will access the Queue and will process the message (retrieving the video or audio input file from the temporary storage, if needed).
5. The SignON Pipeline will prepare the result message and will place it in the Queue.
6. The API Manager will retrieve the message from the Queue and send it back to the SignON App as the RESTful API response to the initial request.

[1] https://min.io/
[2] https://www.rabbitmq.com/

3.3.2. Correction Interaction

A “correction interaction” can occur only after a “standard interaction”. In such a case, the user will be able to provide a correction to the SignON App, which will be sent along with the original input for “permanent” storing (until the appropriate SignON Pipeline component learns the correction, then it will be deleted).

To achieve this result, the following steps are foreseen:

1. The user will authenticate through the SignON App, which will interact with the User Authentication sub-component, through the usage of the RESTful API.
2. The SignON App will send the user’s correction along with the original input and metadata to the API Manager, through the usage of the RESTful API.
3. The API Manager will check the user authentication through the User Authentication sub-component and accept or reject the request.
4. The API Manager will save the provided correction along with the original input and metadata to the Permanent Storage. This will be feasible through the usage of technologies such as MinIO, which will provide a flexible object storage (that can be initially deployed on a private solution, and be eventually migrated to a cloud service provider) until processed by the appropriate SignON Pipeline component, as mentioned above.

Further details about the SignON Orchestrator will be released later in the project and will be presented in the following deliverables:

- *D2.3 - First release of the SignON Open Cloud platform, including the Open Cloud Platform design, due by M13 (January 2022).*
- *D2.4 - Intermediate release of the SignON Open Cloud Platform, due by M26 (February 2023).*
- *D2.5 - Final release of the SignON Open Cloud Platform, due by M36 (December 2023).*

3.4. SignON Pipeline

The SignON Pipeline will be responsible for interpreting the input, translating it and synthesising the output.

In order to accomplish all of the proposed functionalities, the SignON Pipeline will be comprised of three different components (Source Language Processing, Intermediate Representation and Message Synthesis) as shown in the following figure (Figure 4).

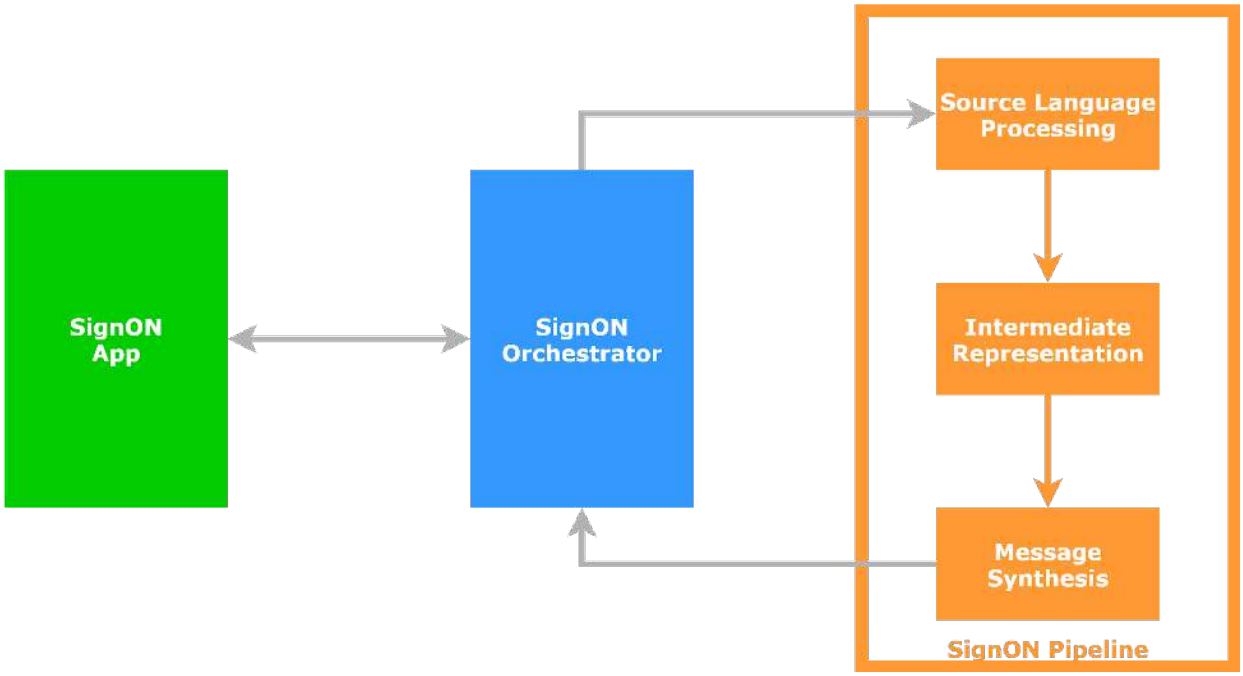

Figure 4 – SignON Pipeline architecture overview.

A high level description of these components is provided in the following subsections, while deep technical details (e.g. the training process of each sub-component, etc.) are left to more specific deliverables mentioned in each subsection.

3.4.1. Source Language Processing

The Source Language Processing component will be responsible for the interpretation of the input provided by the user (video, audio or text). In order to correctly manage each type of input, three dedicated sub-components are present: the Sign Language Recognition (SLR) for video input, the Automatic Speech Recognition (ASR) for audio input and the Natural Language Understanding (NLU) for text input (and transcriptions of audio).

The following figure (Figure 5) shows the internal architecture of the Source Language Processing component.

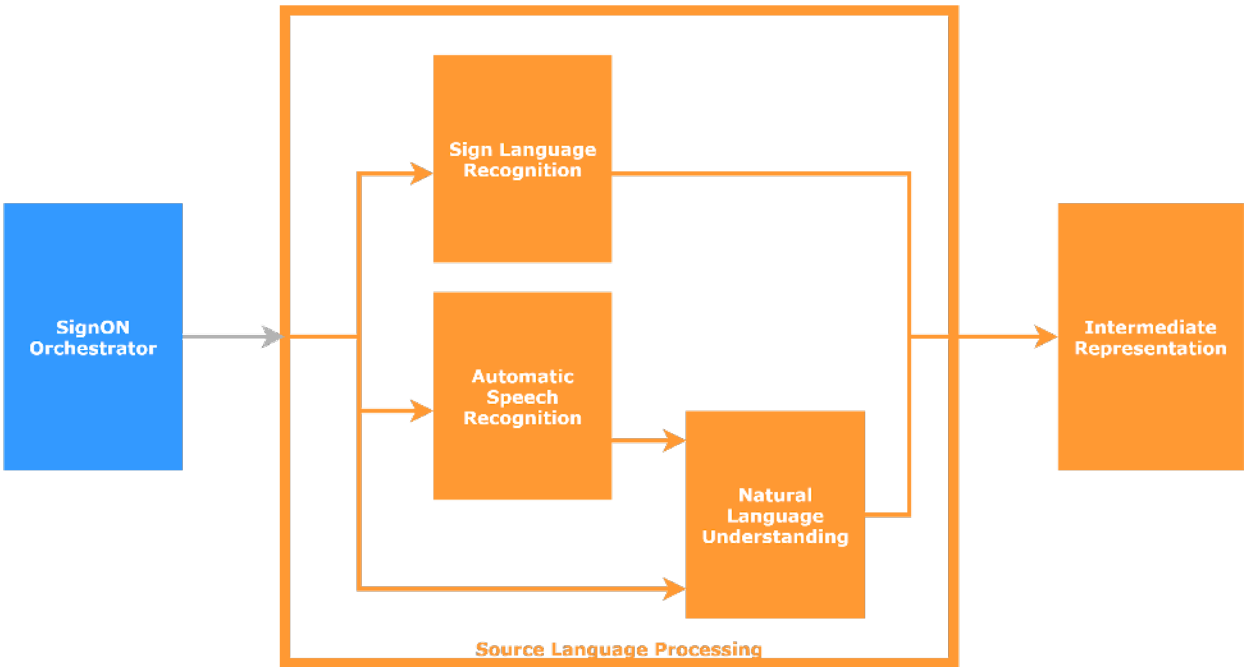

Figure 5 – Source Language Processing architecture overview.

The SLR sub-component will extract a computational representation of what is being signed from sign language videos. This will feed directly into the Intermediate Representation.

The ASR sub-component will convert audio into its textual form in the source language. This text output will be fed to the NLU sub-component.

The NLU sub-component will receive text as input from the SignON user, or transcriptions obtained via the ASR sub-component. This pipeline will perform text normalisation, adaptation and spelling correction. Other natural language processing techniques such as named entity recognition will also be applied. The output of this pipeline will be fed into the Intermediate Representation.

Further details will be released later in the project and will be presented in the following deliverables:

- *D3.1 - Internal repository with language data resources (sign and oral), due by M24 (December 2022)*
- *D3.2 - Sign language recognition component and models, due by M30 (June 2023)*
- *D3.4 - Automatic speech recognition component and models, due by M30 (June 2023)*
- *D3.3 - Linguistic description for ISL, BSL, VGT, NGT and LSE, due by M36 (December 2023)*
- *D3.5 - First Natural Language Processing Pipelines, due by M16 (April 2022)*
- *D3.6 - Second Natural Language Processing Pipelines, due by M36 (December 2023)*

3.4.2. Intermediate Representation

The Intermediate Representation component will be responsible for the translation of the input message in one language into a message in another language.

In order to achieve this result, the following sub-components are proposed: Data Formatting, Context and Metadata Integration, InterL and Quality Assessment and Hybridisation.

The following figure (Figure 6) shows the internal architecture of the Intermediate Representation component.

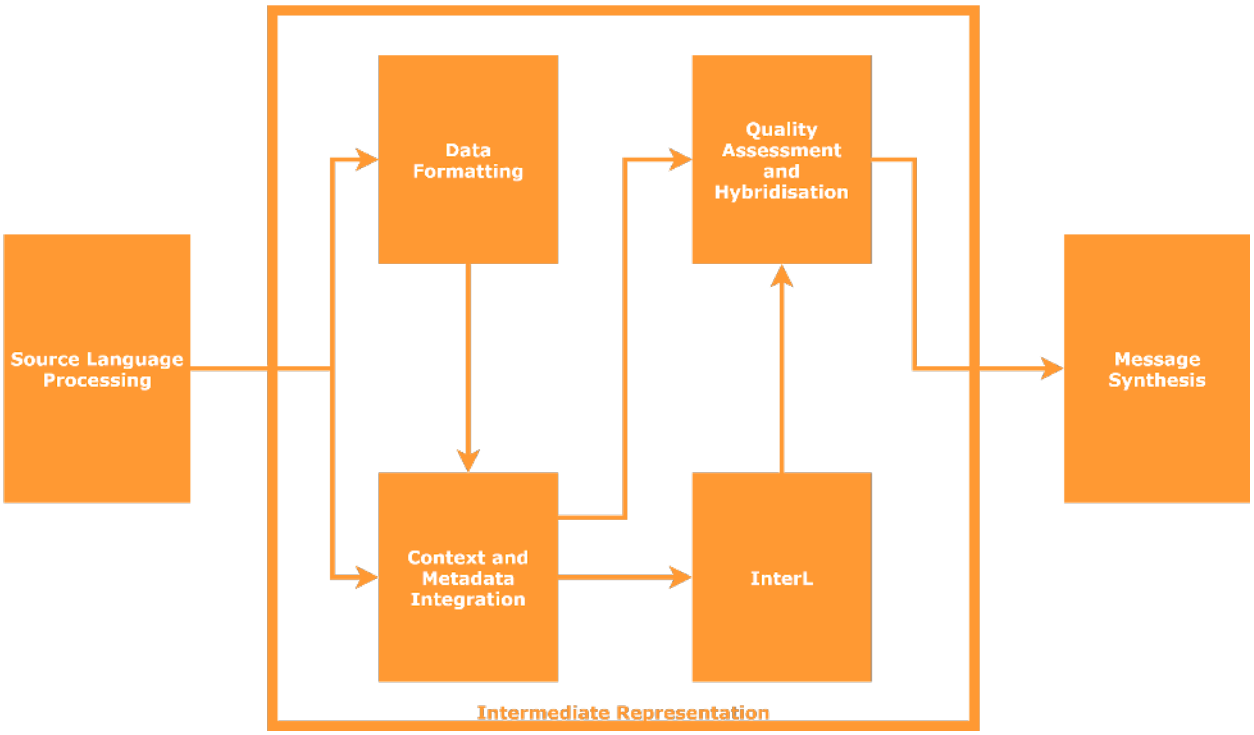

Figure 6 – Intermediate Representation architecture overview.

Once the input message is sent to the Intermediate Representation component, the text or recognised SL will be reformatted if necessary to fit the requirements of the following sub-components. E.g. the text encoding will be checked and converted into UTF-8 encoding if necessary. At this stage the input will be the recognised or typed text or the recognised sign language. The input and the accompanying metadata, e.g. speaker demographics, conversation domain, etc., will be combined into a JSON object that will then be sent to the Inter-L for translation. Note that Inter-L would consist of InterL-E and

InterL-S that will operate independently[3] and their output will be assessed, in terms of quality, and the best translation will be propagated to the next sub-component. The Quality Assessment and Hybridisation sub-component will assess the quality of the translated message and send the translation, along with information about its quality as output to the Message Synthesis component.

Further details can be found in the following submitted deliverables:

- *D4.3 - First distributional intermediate representation based on embeddings - InterL-E*
- *D4.6 - First Routines for transformation of text from and to InterL*
- *D4.11 - First adaptable pipeline for training and updating the InterL*

Other details will be released later in the project and will be presented in the following deliverables:

- *D4.1 - First symbolic intermediate representation - InterL-S, due by M18 (June 2022)*
- *D4.2 - Second symbolic intermediate representation - InterL-S, due by M32 (August 2023)*
- *D4.4 - Second distributional intermediate representation based on embeddings - InterL-E, due by M30 (June 2023)*
- *D4.5 - A hybrid intermediate representation, due by M36 (December 2023)*
- *D4.7 - Second Routines for transformation of text from and to InterL, due by M18 (June 2022)*
- *D4.8 - Final Routines for transformation of text from and to InterL, due by M32 (August 2023)*
- *D4.9 - First Routines for transformation of SL representations from and to the InterL, due by M22 (October 2022)*
- *D4.10 - Final Routines for transformation of SL representations from and to the InterL, due by M32 (August 2023)*
- *D4.12 - Second adaptable pipeline for training and updating the InterL, due by M24 (December 2022)*

[3] This is something that will be researched and investigated in detail in the future.

3.4.3. Message Synthesis

The Message Synthesis component will be responsible for the display to each user of their chosen output text, spoken language and/or sign language (with a 3D virtual signer). In order to achieve this result, the following sub-components are foreseen: Speech / Text Generator, SIGN_A Representation For Synthesis, SIGN_A To BML Planner and Virtual Representation.

The following figure (Figure 7) shows the internal architecture of the Message Synthesis component.

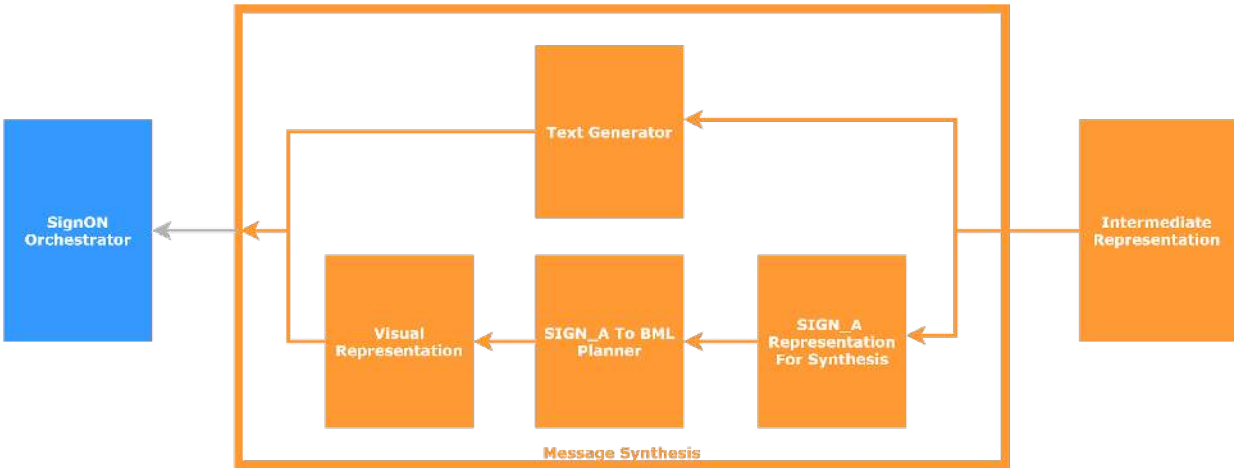

Figure 7 – Message Synthesis architecture overview.

The Text Generator sub-component will receive the Intermediate Representation as input and will generate text based on the message metadata. The generated text will be then fed to the SignON Orchestrator. As Text-To-Speech is meant to be based on a third-party service, it will be called directly from the SignON App.

The SIGN_A Representation for Synthesis sub-component will receive the Intermediate Representation as input and will create a SIGN_A representation that will be then fed to the SIGN_A To BML Planner sub-component.

The SIGN_A To BML Planner sub-component will parse and convert the SIGN_A representation to a BML representation which is useful for avatar animation. This BML will be fed to the Visual Representation sub-component.

The Visual Representation sub-component will use the BML as input to synthesise a set of animations to be performed by the virtual character and then it will create a link to a 3D scene as output that will be

fed to the SignON Orchestrator. This 3D scene will be used later to load the virtual avatar with the appropriate animations into the SignON App.

Further details will be released later in the project and will be presented in the following deliverables:

- *D5.1 - First version of virtual character, due by M18 (June 2022)*
- *D5.2 - Final version of virtual character, due by M36 (December 2023)*
- *D5.3 - Interactive co-creation web-based platform for learning from user input, due by M12 (December 2021)*
- *D5.4 - First Sign language-specific lexicon and structure, due by M8 (August 2021)*
- *D5.5 - Second Sign language-specific lexicon and structure, due by M24 (December 2022)*
- *D5.6 - Final Sign language-specific lexicon and structure, due by M36 (December 2023)*
- *D5.7 - A planner for translating from Sign_A to BML-based script, due by M12 (December 2021)*
- *D5.8 - A realiser of BML-based script to 3D animated character, due by M18 (June 2022)*
- *D5.9 - InterL to Text to speech pipeline, due by M18 (June 2022)*

4. Conclusion

In order to facilitate the exchange of information between DHH and hearing individuals, the SignON project will develop a free, open application and framework for conversion between video (capturing and understanding sign language), audio and text, as well as translation between signed and spoken languages.

This document presents the high level architecture of the SignON Framework, and describes the interfaces, the components, their interactions and their sub-components.

Some implementation details have been only partially covered and will be described in the forthcoming deliverables (mentioned in this document) that will be released later in the project.