Cost effective, Multilingual, Privacy-driven voice-enabled Services

www.compriseh2020.eu

Call: H2020-ICT-2018-2020
Topic: ICT-29-2018
Type of action: RIA
Grant agreement Nº: 825081

WP Nº5:	Cloud-based platform for multilingual voice interaction
Deliverable Nº5.2:	Platform hardware and software architecture
Lead partner:	TILDE
Version Nº:	1.0
Date:	27/11/2019

European Commission

Document information	
Deliverable Nº and title:	D5.2 - Platform hardware and software architecture
Version Nº:	1.0
Lead beneficiary:	Tilde (TILDE)
Author(s):	Askars SALIMBAJEVS (TILDE), Raivis SKADIŅŠ (TILDE)
Reviewers:	Irina ILLINA (INRIA), Gerrit KLASEN (ASCO).
Submission date:	27/11/2019
Due date:	30/11/2019
Type[1]:	R
Dissemination level[2]:	PU

Document history			
Date	Version	Author(s)	Comments
08/11/2019	0.1	Askars SALIMBAJEVS, Raivis SKADIŅŠ	Initial draft version
20/11/2019	0.2	Askars SALIMBAJEVS, Raivis SKADIŅŠ	Final version based on the reviewers' comments
27/11/2019	1.0	Emmanuel Vincent & Zaineb Chelly	Final version reviewed by the Coordinator and the project manager

1 **R**: Report,

2 **CO:** Confidential, only for members of the consortium (including the Commission Services)

Document Summary

This deliverable defines user profiles and stories, requirements, architecture and implementation plan for the COMPRISE Cloud platform.

The platform will allow users to upload, store and manage data and labels, and train or access user-independent domain-specific models. Two types of data will be handled by the platform: (1) speech and (2) text. The platform will allow users to train acoustic and language models for automatic speech recognition (ASR), and intent detection models for natural language understanding (NLU) on collected data. In the future, support for other types of data and models might be added.

The main users of the cloud-based COMPRISE platform will be developers using the COMPRISE SDK (WP4) which will exchange data and models via a REST API. Communications between the platform and the users' devices will be secured via state-of-the-art encryption, and full compliance with the GDPR (e.g., regarding data retention) will be ensured.

This platform will fill a gap in the current ecosystem: existing resource repositories are good for speech resource description, dissemination, sharing, and distribution, but according to our knowledge there is no platform that would facilitate speech data creation, labelling, and curation. The COMPRISE platform will be designed and developed for these purposes.

In order to bootstrap the process when creating resources for a new language and/or a new domain, the cloud-based COMPRISE platform will use machine translation to translate textual training data to the new language.

The platform will also include web-based UI which will allow to sign up to the COMPRISE platform, perform general operations and access the documentation. An initial version of this platform will be setup during the project runtime as a demonstrator.

Table of contents

1. Introduction

The COMPRISE Cloud platform is developed in the scope of Work Package 5 (WP5) "Cloud-based platform for multilingual voice interaction". The objectives of this work package are to:

- bring together the results of WP2, WP3 and WP4 to develop a cloud-based platform to collect users' neutralised speech and text data and curate them.
- provide access to the user-independent speech-to-text and spoken language understanding models trained on these data as a service via a web service API.

The current report is developed in the scope of task T5.1. "Hardware and software architecture" and consists of the following sections:

- **Section I** provides an overview of WP5 objectives and the document structure.
- **Section II** defines the COMPRISE platform user profiles and the corresponding user stories, details both the functional and non-functional requirements for the platform and the web-based UI.
- **Section III** defines the architecture of the COMPRISE platform and details the infrastructure, the main web services, authentication, authorisation, and the proposed API.
- **Section IV** describes the implementation plan for the COMPRISE platform.
- Finally, **Section V** is devoted to conclusions.

2. User requirements

In order to specify user requirements for the COMPRISE Cloud platform first it is necessary to understand who platform users will be and how they will use the platform. Therefore, the following process was implemented:

- First, Cloud platform user profiles were defined.
- Next, for each user profile a list of associated user stories was created.
- Finally, based on these stories, functional and non-functional requirements for the Cloud platform were specified.

2.1. User profiles

2.1.1. Client App

A Client App is an application that uses the COMPRISE SDK Client Libraries for automatic speech recognition (ASR), natural language understanding (NLU), and/or other spoken dialogue components and for communication with the COMPRISE Cloud platform. In order to achieve the best possible user experience, the Client App wants to use the best ASR and/or NLU models for the targeted usage domain. This is achieved by periodically (at runtime): (1) uploading new neutral speech and/or text data to the cloud-based COMPRISE platform, and (2) downloading the latest domain-specific models from the platform (e.g., on application start). The notion of usage domain depends on the component: for ASR acoustic modelling this typically refers to a given speech capture mode (e.g., mobile phone, smart speaker, etc.) and a given language, while for ASR language modelling and NLU this refers to the goal of the application (e.g., drive-thru, cooking support, medical note taking, etc.).

2.1.2. Developer

Developers use the COMPRISE SDK to create voice-enabled privacy-preserving applications (e.g., a personal assistant). In order to achieve the best possible user experience, they want to use domain-specific ASR and/or NLU models for the particular usage domain of the application.

The domain-specific neutral data collected for each application and each language is grouped into separate corpora: speech data is appended to the application's speech corpus, and text data is appended to the application's text corpus. Subject to the user and the Developer consent, these corpora can be merged into larger domains as appropriate, e.g., "French mobile phone speech" (merging all French mobile phone speech corpora collected by various applications to train ASR acoustic models) or "English cooking support" (merging all corpora collected by various cooking support applications to train ASR language models and NLU models).

Developers use the cloud-based COMPRISE platform to manage the collected data, process the collected data (e.g., apply machine translation) and train domain-specific user-independent ASR and/or NLU models. After successful training, the models are downloaded and used by the Client App.

As the collected data must be annotated, the Developers shall be able to give access to the collected corpora to Data annotators employed by the Developer's company.

2.1.3. Data annotator

Data annotators use the cloud-based COMPRISE platform to label domain-specific neutral speech and text data.

They are granted access to speech or text corpora by Developers or by the Administrator. Each Data annotator can have access to multiple corpora simultaneously.

For speech corpora, the Data annotators provide a written transcription of each audio recording. For text corpora, the Data annotators label each user prompt in terms of intent (NLU) or next dialog state (DM). The labelled data is then used for training domain-specific models.

2.1.4. Administrator

The Administrator maintains the COMPRISE Cloud platform and manages global access to its resources by creating, approving and deleting user accounts. Subject to the user and the Developer consent, he/she can also grant access to Data annotators employed by the voice technology company operating the platform in order to label certain data.

2.2. User stories

After user profiles are defined, the next task is to define typical user stories for each user profile.

2.2.1. Client App

ID	DATA_UPLOAD
User	Client App
Action	Upload anonymised speech and text data to the COMPRISE platform
Motivation	Provide in-domain data for training improved ASR/NLU models
Description	Client App uses COMPRISE SDK Client Libraries to perform ASR and/or NLU on data from real users. Client App anonymises speech/text data and after authentication and authorisation uploads it to a reserved place (appends to some corpus) on the COMPRISE platform, where it can be used to train improved in-domain models.

ID	MODEL_DOWNLOAD_API
User	Client App
Action	Download trained ASR/NLU model
Motivation	Use improved ASR/NLU model
Description	Client App uses the COMPRISE SDK to gain access (after authentication and authorisation) to the list of trained models available to the particular Client App on the COMPRISE platform. Client App can choose which ASR/NLU model it wants to download.

2.2.2. Developer

ID	CORPUS_CREATE
User	Developer
Action	Create a reserved space (corpus) where the Client App can upload the in-domain neutral data
Motivation	Collect and group in-domain neutral data in one place
Description	User logs in to the COMPRISE platform. User goes to the list of corpora.

	User adds a new corpus, chooses the corpus type (speech or text) and enters the name of the corpus A new empty corpus is created.

ID	CORPUS EDIT
User	Developer
Action	Edit corpus metadata
Motivation	Update corpus metadata
Description	User logs in to the COMPRISE platform. User goes to the list of corpora he/she has access to. User selects a corpus. User can change corpus metadata.

ID	CORPUS_DELETE
User	Developer
Action	Delete the reserved space (corpus) where the Client App can upload the in-domain neutral data
Motivation	Delete data which is no longer needed
Description	User logs in to the COMPRISE platform. User goes to the list of corpora. User sees only corpora he/she has access to. User selects a corpus. User presses "Delete corpus". User confirms deletion. The corpus is removed from the platform.

ID	CORPUS_BROWSE
User	Developer
Action	Browse the list of corpora, get basic statistics
Motivation	Analyse collected data
Description	User logs in to the COMPRISE platform. User goes to the list of corpora. User sees only corpora he/she has access to. User browses the list. When a corpus is selected, basic information on this corpus is displayed: name, amount of data, time and date of last modification.

ID	CORPUS TRANSLATE
User	Developer
Action	Request machine translation of text corpus
Motivation	Train NLU model for new language
Description	User logs in to the COMPRISE platform. User goes to the list of corpora. User sees only corpora he/she has access to. User selects a corpus. User presses “Translate corpus”, selects language and confirms corpus translation. The platform creates a new corpus and performs translation. When translation is finished, User receives a notification.

ID	MODEL_TRAIN
User	Developer
Action	Create new ASR/NLU model from in-domain neutral data
Motivation	Improve ASR/NLU performance with real world in-domain data
Description	User logs in to the COMPRISE platform. User requests to train a model. User sees the list of corpora he/she has access to. User selects a corpus from the list. The model training job is added to the queue. When model training is finished, User receives a notification.

ID	DEV MODEL DOWNLOAD
User	Developer
Action	Download trained ASR/NLU model
Motivation	Integrate improved ASR/NLU model into application
Description	User logs in to the COMPRISE platform. User goes to the list of trained models he/she has access to. User selects a model and requests download. The model is downloaded.

ID	MODEL DELETE
User	Developer

Action	Delete ASR/NLU model
Motivation	Delete ASR/NLU model from application
Description	User logs in to the COMPRISE platform. User goes to the list of trained models. User sees only the models he/she has access to. User selects a model and requests to delete it. System asks for confirmation. The model is deleted.

ID	SET_API_KEY
User	Developer
Action	Set access credentials for Client App
Motivation	Client App needs credentials to access the COMPRISE platform API.
Description	User logs in to the COMPRISE platform. User can create new access credentials for the API. The created credentials can be built into some Client App. Old API credentials are automatically invalid.

2.2.3. Data annotator

ID	TEXT_DATA_ANNOTATION
User	Data annotator
Action	Text data annotation for NLU
Motivation	Neutral in-domain text data shall be adapted to be used in training
Description	User logs in to the COMPRISE platform. User goes to the list of corpora. User sees only corpora he/she has access to. User selects a corpus and presses “Annotate corpus”. User sees the list of (text) utterances. User can add/edit labels for each utterance in the corpus. User can correct each utterance in the corpus. User can remove utterances from the corpus.

ID	SPEECH_DATA_ANNOTATION
User	Data annotator
Action	Speech data annotation for ASR

Motivation	Neutral in-domain speech data shall be adapted to be used in training
Description	User logs in to the COMPRISE platform. User goes to the list of corpora he/she has access to. User selects a corpus and presses “Annotate corpus”. User sees the list of audio recordings. User can listen to each audio recording. User can add/edit the transcription for each audio recording. User can remove audio recordings from the corpus.

2.2.4. Administrator

ID	CREATE_DEVELOPER
User	Administrator
Action	Create new Developer account
Motivation	Give access to COMPRISE platform to some Developer or organisation
Description	User logs in to the COMPRISE platform. User goes to the list of Developers/organisations. User creates a new Developer/organisation account by providing name, e-mail and other information.

ID	EDIT DEVELOPER
User	Administrator
Action	Edit Developer account
Motivation	Make corrections to the Developer account
Description	User logs in to the COMPRISE platform. User goes to the list of Developer accounts. User selects a user from the list and presses “Edit”. User can change group, name, email or other information.

ID	DELETE_DEVELOPER
User	Administrator
Action	Delete Developer account
Motivation	Revoke access to the COMPRISE platform
Description	User logs in to the COMPRISE platform. User goes to the list of Developer accounts.

	User selects a Developer from the list and presses “Delete”. User confirms deletion. Account is deleted from the COMPRISE platform and can no longer be used to access it. Resources associated with this account are deleted from the platform.

ID	ADM_MODEL_DELETE
User	Administrator
Action	Delete ASR/NLU model
Motivation	Delete ASR/NLU model from application
Description	User logs in to the COMPRISE platform. User goes to the list of trained models. User sees all models stored on the platform. User selects a model and requests to delete it. System asks for confirmation. Model is deleted.

ID	ADM_CORPUS_DELETE
User	Administrator
Action	Delete a reserved space (corpus) where Client App can upload the in-domain neutral data
Motivation	Delete data which is no longer needed
Description	User logs in to the COMPRISE platform. User goes to the list of corpora. User sees all corpora stored on the platform. User selects a corpus. User presses “Delete corpus”. User confirms deletion. The corpus is removed from the platform.

2.3. Requirements

2.3.1. Platform requirements

ID	Requirement
FUN-001	The COMPRISE platform shall provide a complete system to collect and annotate neutral speech and text data, train ASR/NLU models that address the needs of:

ID	Requirement
	● App Developers ● Data annotators ● Speech and NLU Scientists ● Administrators
FUN-002	The COMPRISE platform shall implement multiple-level access control for the API clients: ● Apps are only allowed to access non-destructive API methods, i.e., data upload and model download. ● Data annotators are only allowed to annotate the corpora they have been assigned. ● Developers have access to all functions of the platform except user management. ● Administrators have access to all functions including user management. All clients are allowed to access only resources for which they have authorisation (they own the resource or have a key for it).
FUN-003	The COMPRISE platform shall handle (store, edit, access for training, delete) speech and text corpora for various domains.
FUN-004	The COMPRISE platform shall handle ASR and NLU models (store, train, retrieve, delete) for various domains.
FUN-005	The COMPRISE platform shall provide a user interface to allow users to access functionalities.
FUN-006	The COMPRISE platform shall prompt the user prior to performing any operation that cannot be undone.
FUN-007	The COMPRISE platform shall allow the annotation of speech corpora.
FUN-008	The COMPRISE platform shall allow the annotation of text corpora.
FUN-009	The COMPRISE platform shall handle (store, assign roles, delete) user accounts.
FUN-010	The COMPRISE platform shall handle (create, store, revoke) API access keys.
FUN-011	The COMPRISE platform shall allow Client Apps to upload neutral speech and text data and append them to the specified corpus.
FUN-012	The COMPRISE platform shall be able to train ASR models using a scalable architecture able to handle big data sets and high computational load.

ID	Requirement
FUN-013	The COMPRISE platform shall be able to train NLU models using a scalable architecture able to handle big data sets and high computational load.
FUN-014	ASR and NLU training shall be done using training tools that will be developed in WP2 - T2.3.
FUN-015	The COMPRISE platform shall allow clients to download trained ASR/NLU models.
FUN-016	The COMPRISE platform shall be able to perform machine translation of text data using a cost-effective scalable service able to handle big data sets and high computational load.

2.3.2. Web-based UI requirements

ID	Requirement
FUN-017	The user interface shall be a web-based access point for the COMPRISE platform.
FUN-018	The user interface shall allow multiple instances to run simultaneously.
FUN-019	The user interface shall provide role-based access control for the following categories of authorised users: ● Developers ● Data annotators ● Administrators
FUN-020	The user interface shall be compatible with at least the following web browsers: ● Microsoft Edge version 21 onwards ● Firefox from version 58 onwards ● Google Chrome from version 64 onwards ● Apple Safari 11 onwards
FUN-021	The user interface shall provide a section where authorised users can create a new corpus for ASR or NLU.
FUN-022	The user interface shall provide a section where authorised users can access the created corpus to view/edit metadata.
FUN-023	The user interface shall provide a section where authorised users (Developers or Administrators) can delete whole corpora.
FUN-024	The user interface shall provide a section where authorised users can trigger ASR/NLU model training.

ID	Requirement
FUN-025	The user interface shall provide a section where authorised users can trigger text corpus machine translation.
FUN-026	The user interface shall provide a section where authorised annotators can annotate speech recordings.
FUN-027	The user interface shall provide a section where authorised users can annotate text data.
FUN-028	The user interface shall provide a section where Developers or Administrators can control annotator access to corpora.
FUN-029	The user interface shall provide a section where Administrators can create/delete Developer accounts.
FUN-030	The user interface shall provide a section where Developers can set API access credentials for apps.
FUN-031	The user interface shall provide a section where Developers can download trained ASR/NLU models.
FUN-032	The user interface shall provide sections where authorised Developers can view lists of corpora, models.
FUN-033	All sections where a list of elements is displayed will include a search input text control.
FUN-034	The user interface shall have a public main page that is accessible without authorisation. This page should contain information about COMPRISE and a textual description of the registration procedure to become an authorised user of the COMPRISE platform.

2.3.3. Non-functional requirements

ID	Requirement
NF-001	The COMPRISE platform and the web-based user interface shall communicate via web services.
NF-002	The COMPRISE Cloud platform shall ensure full compliance with the GDPR (e.g., regarding data retention) using guidelines and recommendations from D5.1.
NF-003	Communications between the platform and the users' devices shall be secured via encryption.

NF-004	The platform shall be scalable. Idle hardware resources shall be shut down and additional hardware resources shall be requested on demand.
NF-005	Initial deployment and later updates of the platform shall be performed by an automatic process.
NF-006	The COMPRISE platform shall require authentication and authorisation from a user to access platform functionalities.

3. Architecture

3.1. Infrastructure

The COMPRISE platform is expected to work in a cloud environment as a collection of web-services using containerisation (e.g. Docker[3], Kubernetes[4]). Therefore, hardware and system software management will be greatly simplified.

As seen in Figure 1, the COMPRISE platform will consist of 5 main services:

- The Authentication service will authenticate users using the standard OpenID Connect protocol. Since there exist many high-quality authentication solutions and providers, a new solution will not be implemented in the scope of the project. Instead existing authentication solutions or service providers (like Azure B2C) will be utilised.
- The API service will provide COMPRISE platform functionality through an API. The service will be implemented in the scope of the project.
- The Storage service will provide object storage through a web service. Existing cloud storage solutions like Amazon S3 will be utilised as they provide scalability, high availability, low latency, durability and do not require hardware administration. The implementation of an independent storage service will be also considered.
- The Training service will provide training of ASR and NLU models. The service will be implemented in the scope of the project and use model training modules from the partners. For machine translation of the training data the service will use the external Tilde MT machine translation service.
- The web UI will provide a user interface for general COMPRISE platforms functions like registering applications, corpus annotation, triggering model training, etc.

[3] https://www.docker.com/
[4] https://kubernetes.io/

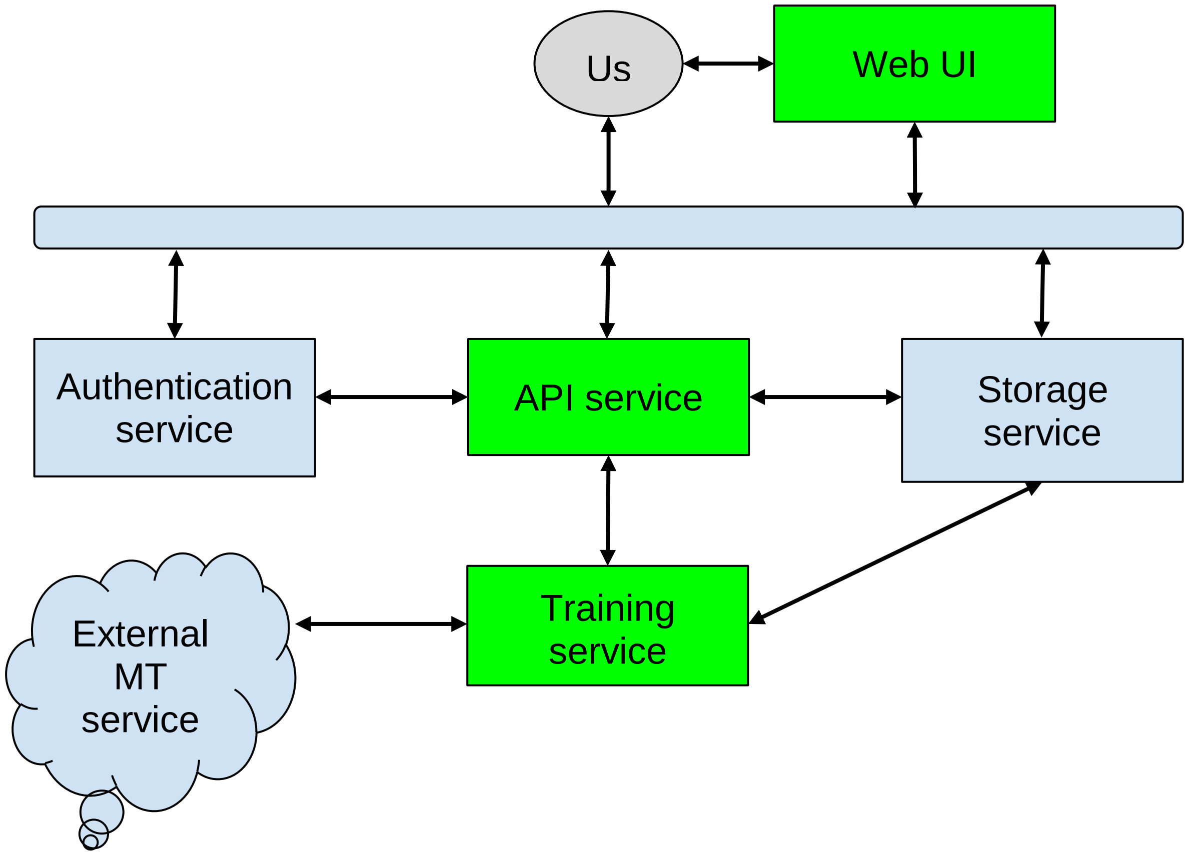

Figure 1. COMPRISE platform services

In order to efficiently balance the load between services and avoid unnecessary resource consumption, COMPRISE platform API clients will request a special upload URL from the API service, which will allow the upload of data to the Storage service directly without the API service acting as intermediary.

All services will be deployable as Docker containers which will allow them to run on almost any cloud provider infrastructure. For Docker container orchestration, we are planning to use Kubernetes.

Kubernetes scheduler and Horizontal Pod Autoscaler (HPA) will be used to run containers only when they are requested and scale to multiple replicas when needed.

An optional gateway or proxy can be used for load-balancing, network administration and protection.

Kubernetes cluster autoscaler will be employed to optimise cluster usage and start additional nodes only when needed.

3.2. Training service

The Training service is responsible for training in-domain user-independent models for ASR and NLU using the model training modules provided by the COMPRISE partners. These modules will be packaged as Docker containers.

The training service will not be exposed to the outside and will be available only inside the cluster. It will receive training requests from the API service and initiate model training by starting training containers as a Kubernetes job. The started containers will have direct access to the training data and models in the Storage service.

Such approach makes it possible to run very different training workloads, improves portability and simplifies dependency maintenance (dependencies and environment are maintained inside containers).

The limitation is that it does not allow traditional distributed training on multiple machines. We plan that in future this limitation can be lifted by using one or both of the following solutions:

- Model training containers can call the Training service API to initiate sub-tasks.
- The Training service can be extended to submit jobs to a classic High Performance Cluster (HPC).

Also, in the future, submission of training jobs to external entities like the European Language Grid[5] will be considered.

For machine translation of the training data, the service will use the external Tilde MT machine translation service. In the future, support for other MT providers can be integrated.

3.3. API service

This service is responsible for providing COMPRISE platform functionality through an API. The service will be exposed to the outside world via an optional gateway or proxy. It would be possible to run multiple API service replicas to deal with high volume of requests.

The API service will be written in Python using the Tornado framework which provides a non-blocking network I/O and theoretically can support tens of thousands of simultaneous connections.

Because each Client App that will use COMPRISE SDK Client Libraries and the COMPRISE platform represents some particular usage domain for which we want to collect data and train models, it was decided to merge the concepts of corpus and application.

3.3.1. Proposed API

The proposed API is based on the REST paradigm.

Application entities are managed using the following REST methods:

[5] https://www.european-language-grid.eu/

Method and URL	Description
GET /applications	Retrieve the list of applications to which the user has access. Returns: JSON, list of ids and names.
GET /applications/<id>	Get data for application <id>. Returns: application JSON (see below)
POST /applications	Create a new application, request body contains application name and other metadata. Input: form.
PUT /applications/<id>	Update data for application <id>. Input: application JSON (see below).
DELETE /applications/<id>	Remove application <id>.

Application JSON is a dictionary that contains:

- string:name - name of the application
- guid:owner - owner/creator of the application
- string:appKey - access key for client apps
- string:speechUploadUrl - URL where to upload speech data
- string:textUploadUrl - URL where to upload text data
- string:annotatorKey - access key for Data Annotators
- string:language - language of the application (ISO 639-1)
- string:description - optional description

Both upload URLs will be unique for each application and contain some key to provide direct access for data upload to the storage service. This allows clients to send in-domain neutral speech and text directly to the Storage service, avoiding a possible bottleneck in the API service.

Two different types of data are collected: speech and text.

Method and URL	Description
GET /applications/<id>/speech	Retrieve list of speech segments in the speech corpus of application <id>. Returns: JSON.

Method and URL	Description
GET /applications/<id>/speech/<utt_id>	Download speech segment <utt_id> or annotation (specified by query parameter). Returns: audio file or annotation.
POST /applications/<id>/speech	Upload speech audio (neutral in-domain data) to speech corpus of application <id>. Input: multipart/form-data.
DELETE /applications/<id>/speech	Delete collection of utterances from application <id> speech corpus. Optional input: list of <utt_id> to delete. If no list is specified, the whole corpus is deleted.
PATCH /applications/<id>/speech/<utt_id>	Submit transcription for speech segment <utt_id>. Input: JSON (format TBD).
DELETE /applications/<id>/speech/<utt_id>	Delete speech segment <utt_id>.
GET /applications/<id>/text	Retrieve list of text segments (utterances) in text corpus of application <id> Returns: JSON.
GET /applications/<id>/text/<id>	Get text segment <utt_id> with annotation Returns: JSON.
POST /applications/<id>/text	Upload text (neutral in-domain data) to text corpus of application <id> Input: multipart/form-data.
DELETE /applications/<id>/text	Delete collection of text segments from application <id> text corpus Optional input: list of <utt_id> to delete. If no list is specified, the whole corpus is deleted.
PATCH /applications/<id>/text/<utt_id>	Annotate text segment <utt_id> Input: JSON.

Method and URL	Description
DELETE /applications/<id>/text/<utt_id>	Delete text segment <utt_id>

The API can be used to access models specific to a given application:

Method and URL	Description
GET /applications/<id>/models	Retrieve the list of model types. Returns: JSON list of model types, e.g. [“ASR”, “NLU”].
GET /applications/<id>/models/ASR	Retrieve list of ASR models for application <id>. Returns: JSON document containing ASR model metadata (see below).
GET /applications/<id>/models/ASR/<m>	Download ASR model <m>. Returns: binary model file.
POST /applications/<id>/models/ASR	Train new ASR model using corpus “/applications/<id>/speech”.
DELETE /applications/<id>/models/ASR</m>	Delete ASR model <m>.
GET /applications/<id>/models/NLU	Retrieve list of NLU models for application <id>. Returns: JSON document containing NLU model metadata (see below).
GET /applications/<id>/models/NLU/<m>	Download NLU model <m>. Returns: binary model file.

Method and URL	Description
POST /applications/<id>/models/NLU	Train new NLU model using one of the following corpora: 1) corpus "/applications/<id>/text" 2) machine translated corpus "/applications/<src>/text" In the second case application id <src> and target language should be provided inside request. NB: Application <src> must belong to the same user as application <id>.
DELETE /applications/<id>/models/NLU/<m>	Delete NLU model <m>.

Both ASR and NLU model JSON contain the following metadata:

- date:created - model creation date
- date:trained - model training date
- bool:latest - is this model latest?
- string:status - is this model trained?
- bool:is_mt - is this model trained on MT data.
- string:recipe - name of recipe used to train a model (TBD when training module development is finished).

The API can also be used to train and access models trained on larger domains comprising data collected from several applications in a similar fashion.

3.3.2. Authentication and authorisation

As mentioned in FUN-2, the API service will implement multiple levels of access for the API. These access levels are implemented by having two types of authentication mechanisms:

- OpenID Connect authentication for Developers and Administrators using an external authentication service.
- API key authentication for Client Apps and Data annotators.

Depending on the authentication method, access to different API methods is provided.

Platform Administrator accounts are created in the OpenID Connect service by the platform owner.

In order to gain access to the API, Developers first sign up using the OpenID Connect service and wait for approval by platform Administrators. Account approval is done inside the OpenID Connect service.

After approval, Developers that are authenticated by OpenID Connect have access to all API methods for resources that they have created. Resources created by other accounts are not visible nor accessible.

Two types of API keys exist in the platform:

- AppKey which is used by Client Apps to access limited platform functionality;
- AnnotatorKey which is used by Annotators to label collected data.

When the client is authenticated with API key X, the following rules apply:

- if X matches the AppKey of application <id>, then the client can use methods:
  - GET /applications/<id>
  - POST /applications/<id>/speech
  - POST /applications/<id>/text
  - GET /applications/<id>/models
  - GET /applications/<id>/models/ASR
  - GET /applications/<id>/models/ASR/<model>
  - GET /applications/<id>/models/NLU/<model>
- if X matches the AnnotatorKey of application <id>, then the client can use methods:
  - GET /applications/<id>/speech
  - PATCH /applications/<id>/speech/<utt_id>
  - DELETE /applications/<id>/speech/<utt_id>
  - GET /applications/<id>/text
  - PATCH /applications/<id>/text/<utt_id>
  - DELETE /applications/<id>/text/<utt_id>.

3.4. Web-based UI

The web-based UI will provide a user interface for general COMPRISE platforms functions to fulfil the requirements specified in Section 2.3.2.

It will be implemented using an Angular web framework and packaged as a Docker container as other services. It can be run directly in the cloud without an explicit virtual machine using services like Azure AppService or in the same Kubernetes cluster as the other COMPRISE platform services.

The web-based UI will allow Developers to sign-up, register applications, access API documentation and try-out forms. An important feature of the web-based UI will be an interface for speech and text data annotation. This interface will be available without creating user accounts using a special URL with embedded Annotator key, which will be created by the Developer and shared with Data annotators.

4. Implementation Plan

As the platform hardware and software architecture has been specified in this deliverable, the next task will be to implement the platform in the scope of T5.2. As this will be a software development task it will include all typical software development activities like setting up environments, coding, testing, automation scripts, deployment scripts, etc. The main challenges of this task will be as follows:

- Implementation of secure cloud-based resource storage;
- Implementation of centralised learning methods, including the implementation of a highly scalable and dynamic cloud-based high-performance computing cluster (model training will require high computing power, but it will be needed occasionally when model retraining will be done);
- Implementation of an API to access the platform functionality from the COMPRISE SDK;
- Implementation of an API to download the trained models for local usage on the user's device;
- Implementation of the strict data protection procedures set in D5.1.

The leading partner will be TILDE; other partners will participate in unit testing, evaluation and integration testing.

Integration will be an important aspect of the task because components that will be developed in WP2, WP3 and WP4 will be included in the platform or will use it through the API.

The approach of continuous implementation will be introduced with agile project management procedures that ensure that major objectives and general tasks are established, controlled and adjusted according to progress.

The development process in Task 5.2 will be divided into 2-week iterations. Each iteration will consist of several major steps – envisioning, planning, implementation and review of key results and uncompleted tasks will be revised, and issues will be identified. This methodology will ensure that the implementation process can be adjusted or re-prioritised based on the results of evaluation.

There will be a project backlog that will contain all platform requirements, features that must be implemented, tasks to be done, and unresolved issues. Backlog items will be prioritised according to their deadlines, implementation costs and importance.

The project backlog will be used to envision and plan iterations. Items from the backlog will be brought to the iteration plan and will be implemented during the iteration. The backlog will be regularly updated at the end of each iteration depending on results achieved during the iteration.

There will be development, and testing environments provided and controlled by TILDE, and there will be acceptance and production environments that will be accessible also to project partners. All development tasks will be done in the development environment, and the system will be regularly (usually daily) deployed to the testing environment to test it. Once the system has been tested, it will be deployed in the acceptance environment where it will be accessible also by project partners. We would like to deploy the system in the acceptance environment at the end of each iteration. Once accepted by other project partners the updated system will be deployed in a production environment.

Although platform development will be done using an agile software development approach, and detailed development plans will be set at the beginning of each iteration, we have a preliminary high-level plan that outlines the main milestones in the development of the platform.

Month	**Date**	**Development milestone**
M12	November, 2019	D5.2. Platform hardware and software architecture prepared; Task 5.2 can be started
M13	December, 2019	Stub API implemented, documented and try-out forms ready
M14	January, 2020	Data upload functionality available
M15	February, 2020	Data management functionality available
M16	March, 2020	First model training functionality available
M17	April, 2020	First Web UI available
M18	May, 2020	All major functionality available, Task 5.4 started
M19	June, 2020	Advanced user management functionality ready
M20	July, 2020	Monitoring and analytics functionality available
M21	August, 2020	All platform functionality implemented

After M21 the work on the platform will continue in the scope of task T5.4. During M22-M30 the platform will be populated with speech and dialog data and tested in several use cases (see WP6). Necessary improvements and corrections will be identified and implemented.

5. Conclusion

This deliverable presented the hardware and software architectures of the cloud-based platform that are being developed within COMPRISE.

In Section 2 of the deliverable, user profiles and associated user stories were defined. Based on these, the functional and non-functional requirements for platform and web-based UI were summarised. The second half of the deliverable described the COMPRISE platform architecture that will be a cloud-based solution consisting of several web services.

To focus on the COMPRISE unique core functionality, existing services and solutions will be used, where possible, for generic platform functions like authentication, storage, machine translation etc. The COMPRISE functionality will be accessible via a REST API, the first version of which is drafted in this deliverable.

Finally, a COMPRISE platform implementation plan was described in the last section of the deliverable.