Funded by the Horizon 2020 Framework Programme of the European Union
ALAMEDA - Grant Agreement 101017558

Deliverable D5.3
Title: ALAMEDA Toolkit 1.0

Dissemination Level:	**PU**
Nature of the Deliverable:	**DEM**
Date:	**30/06/22**
Distribution:	**WP5**
Editors:	**CTL**
Reviewers:	**CERTH, UNISEL**
Contributors:	**ENO, NTNU, UNIC, UPB**

Abstract: This report and the supplementary demonstrator deliverable D5.3 "ALAMEDA Toolkit 1.0" constitute an output from T5.3 and present the first iteration of the AI Toolkit. The toolkit's aim is to host AI-powered modules for monitoring and assessing neurological and brain disorders relevant to Parkinson's, Multiple Sclerosis, and Stroke, in a form that will facilitate their reuse by third party stakeholders, practitioners, and researchers. This report also presents the latest work within T5.4 "Platform Continuous Integration", which is aimed at defining and planning the necessary steps in order to implement the ALAMEDA AI Toolkit and the ALAMEDA platform that will enable the deployment of the research results to the pilot sites where the individual ALAMEDA components will be integrated.

Disclaimer

Revision History

Date	*Rev.*	*Description*	*Partner(s)*
27-May-2022	*0.1*	*Deliverable outline.*	*CTL*
03-June-2022	*0.2*	*Draft of Chapter 2 ready.*	*CTL*
06-June-2022	*0.3*	*Contributions to Chapters 2 & 3 (CSAT, GA, VarCls).*	*CTL, UNIC, NTNU, UPB*
14-June-2022	*0.4*	*Draft of Chapters 1& 4 ready; contributions to Chapter 3.*	*UNISEL, NTNU, CTL*
17-June-2022	*0.5*	*Pre-final version submitted for internal review.*	*CTL*
24-June-2022	*0.6*	*Integrated feedback from internal reviewers; updates to Chapter 3.*	*CTL, ENO, UNISEL, CERTH*
30-June-2022	*1.0*	*Final version ready for submission.*	*CTL*

List of Authors

Partner	*Author*
CTL	*Efstratios Kontopoulos, Christina Michailidou, Konstantinos Avgerinakis*
ENO	*George Koutalieris, Paraskevi Rapti, Harris Georgiou, Iphigenia Kapsomenaki*
NTNU	*Muhammad Sajjad, Adane Nega Tarekegn, Faouzi Alaya Cheikh*
UNIC	*Ioannis Katakis, Pantelis Agathangelou, Chloe Chira*
UNISEL	*Ilias Aliferis, Athanasios Fameliaris, Lora Sasson*
UPB	*Alexandru Sorici*

Table of Contents

Index of Figures

Index of Tables

Glossary

Abbreviation	*Full name*
ADL	*Activities of Daily Living*
AIH	*ALAMEDA Innovation Hub*
API	*Application Programming Interface*
CI/CD	*Continuous Integration and Continuous Delivery*
DL	*Deep Learning*
DoA	*Description of Action*
FE	*Facial Expression*
ML	*Machine Learning*
ND	*Neurological Disorder*
PMSS	*Parkinson's, Multiple Sclerosis, and Stroke*
PROs	*Patient Reported Outcomes*
UI	*User Interface*
WP	*Work Package*

Executive Summary

This report and the supplementary demonstrator deliverable D5.3 “ALAMEDA Toolkit 1.0” constitute an output from T5.3 and present the first iteration of the AI Toolkit. The toolkit’s aim is to host AI-powered modules for monitoring and assessing neurological and brain disorders relevant to Parkinson’s, Multiple Sclerosis, and Stroke, in a form that will facilitate their reuse by third party stakeholders, practitioners, and researchers. The first version of the toolkit is available at https://alameda.catalink.eu/. This report also presents the latest work within T5.4 “Platform Continuous Integration”, which is aimed at defining and planning the necessary steps in order to implement the ALAMEDA AI Toolkit and the ALAMEDA platform that will enable the deployment of the research results to the pilot sites where the individual ALAMEDA components will be integrated.

1. Introduction

This report is supplementary to the demonstrator deliverable D5.3 "ALAMEDA Toolkit 1.0" presenting the first iteration of the AI Toolkit, which is the focus of T5.3. Based on the design and development methodology specified in D5.1, the toolkit hosts the AI-powered modules from WPs 3 and 4 for neurological and brain disorders relevant to PMSS (Parkinson's, Multiple Sclerosis and Stroke), which will, in turn, be published by the Digital Healthcare Innovation Hub in WP7. Fig. 1 illustrates diagrammatically the interplay between the AI Toolkit and the rest of the relevant tasks and WPs.

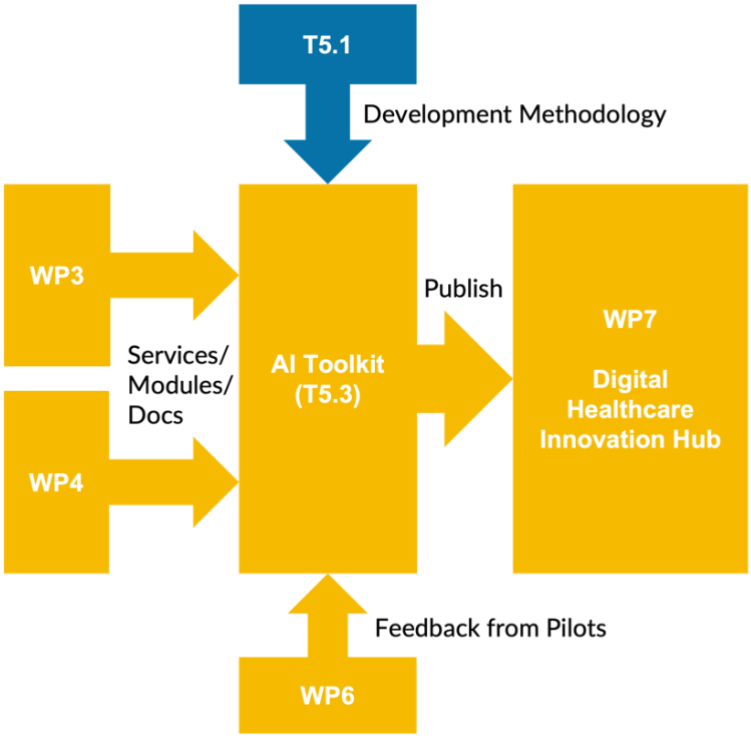

Fig. 1. Interrelationship between the AI Toolkit and the other ALAMEDA WPs and tasks.

D5.3 presents the first development cycle for the toolkit, while the second and final iteration, which will be presented in D5.5 (due M30) will be largely based on feedback that will be collected from the project pilots (WP6). The first version of the toolkit is available at https://alameda.catalink.eu/

Besides the AI Toolkit, this report also presents the latest work within T5.4 "Platform Continuous Integration", which is aimed at defining and planning the necessary steps in order to implement the ALAMEDA AI Toolkit and the ALAMEDA platform that will enable the deployment of the research results to the pilot sites where the individual ALAMEDA components will be integrated. According to the Description of Action, the latter task has no dedicated deliverable to present its outcomes; therefore, deliverables D5.3 (the present report) and D5.5 "ALAMEDA Toolkit 2.0" (due M30) will serve the purposes of presenting these results.

The rest of this report is structured as follows:

- **Chapter 2** presents the approach and methodology underlying the design and development of the AI Toolkit v1.
- **Chapter 3** presents in detail the first version of the AI Toolkit and the modules encompassing it.
- **Chapter 4** presents the adopted approach for the AI toolkit and ALAMEDA platform integration.
- **Chapter 5** concludes this report and gives directions for the next steps leading to the AI Toolkit v2.

2. Approach & Methodology

The AI Toolkit aspires to constitute the "gateway" for interested third-party stakeholders and the overall research and healthcare community wishing to use the ALAMEDA AI-powered services developed within WP3 and WP4, with a special focus on the care of patients with brain disorders. The toolkit takes the form of a web-based platform hosting the AI-enabled modules along with the accompanying documentation and resources and will ultimately be available through the ALAMEDA Innovation Hub (WP7). This chapter presents the approach and methodology underlying the design and development of the AI Toolkit v1.

2.1 Design Approach

According to the Description of Action (DoA) and deliverable D5.1 "Description Methodology & Design Principles of ALAMEDA AI Toolkit Version 1.0", the AI Toolkit will include core functional modules of applications developed within WPs 3 and 4. This means that the toolkit modules do not encompass "application-level" features, like, e.g., user authentication, User Interface (UI), etc., but, instead, feature a core input-output operation via respective Application Programming Interfaces (APIs). This way, a rich toolset of open-source AI-powered modules is available to third parties, to base their applications on.

As specified in D5.1, the following modules (listed alphabetically) are included in the first version of the AI Toolkit, with the potential of adding more modules in the second and final version (due M30):

- **Conversation Sentiment Analysis Toolkit** (WP4, partner: UNIC)
- **Facial Emotion Recognition Toolkit** (WP4, partner: NTNU)
- **Gait Analysis Toolkit** (WP4, partner: NTNU)
- **Predictor Variable Time Series Classification** (WP3, partner: UPB)
- **Semantic Knowledge Graph** (WP4, partners: CTL & CERTH)
- **Sleep Monitoring and Assessment** (WP4, partner: ENO)

Every module is accompanied by a rich set of resources, API specifications, relevant datasets, tutorials, and instructions for installation/deployment and execution, in order to ensure a smooth learning curve and facilitate the rapid and effortless use of the ALAMEDA tools.

2.2 Design and Development Methodology

Immediately after the submission of D5.1, we kickstarted an agile approach for designing and developing the AI Toolkit, with biweekly "sprints" culminating in group telcos, where progress was reported and firm action points for the next sprint were decided. During each sprint, the task leader CTL had frequent bilateral communication with the partners developing the modules, in order to monitor progress and coordinate their efforts in a uniform fashion.

2.2.1 Internal Milestones

During the kickstarting phase of the agile approach, we specified the following task-specific (internal) milestones, until the release of the AI Toolkit v1 in M18:

Milestone #1 – API Specification: This milestone entailed initially the specification of a draft of each module's API, followed by a formal specification via the OpenAPI standard[1]. In order to not burden the partners with getting familiarized with the OpenAPI standard themselves, partner CTL co-created draft specs in the form of tables together with the partners, and, after those were finalized, the task leader developed the specifications according to the standard. Fig. 2 illustrates the draft API specification for partner UNIC's Conversation Sentiment Analysis Toolkit (CSAT) as an example, while the OpenAPI specifications for all components are included in the AI Toolkit v1, presented in the next chapter.

Endpoint name	Sentiment [CSAT-01]
Endpoint description	Retrieve the sentiment class with the highest score for a given piece of text.
Endpoint URL	/csat/sentiment
HTTP method	POST
Request parameters	N/A

Request body	{
	"text": "I had a lovely walk in the park today."
	}

Sample request	/csat/sentiment

Response body	{
	"sentiment_class": "positive",
	"sentiment_score": 70
	}

Endpoint name	Sentiment classes [CSAT-02]
Endpoint description	Retrieve the scores of each sentiment class for a given piece of text.
Endpoint URL	/csat/classes
HTTP method	POST
Request parameters	N/A

Request body	{
	"text": "I had a lovely walk in the park today."
	}

Sample request	/csat/classes

Response body	{
	"sentiment_classes": [
	{
	"sentiment_class": "positive",
	"sentiment_score": 70
	},
	{
	"sentiment_class": "neutral",
	"sentiment_score": 25
	},
	{
	"sentiment_class": "negative",
	"sentiment_score": 5
	}
	]
	}

Fig. 2. Draft API specification for UNIC's CSAT module.

Milestone #2 – Module Documentation: The second milestone involved the textual descriptions and supplementary material for each module, which would be included in the module's respective dedicated page on the toolkit. The documentation included the following:

- module title & acronym,
- partner(s) responsible,
- 1-sentence description,
- longer description,
- inputs and outputs,
- background,
- installation instructions,
- datasets & samples.

Additional optional fields were of course allowed, as per each partner's request.

[1] https://www.openapis.org/

Milestone #3 – Module Deployment: The third milestone assessed the successful deployment of the toolkit modules, with a focus on having a fully operational AI Toolkit by M18. To this end, a container registry was established in the project's GitLab infrastructure. Subsequently, module developers were instructed to build and push their components to the registry as re-usable Docker images that will be available for third-party stakeholders. A guest account was setup to allow read-only access to the registry, in order to allow pulling the images of interest, using instructions provided by the AI toolkit.

Milestone #4 – Module Testing: The final milestone was responsible for final testing to the modules, towards having a fully operational AI Toolkit for demonstration by M18.

2.2.2 Website Design

The design of the website hosting the AI Toolkit is very simple, consisting only of a welcome page that briefly presents each tool, and dedicated sub-pages for each tool and for the accompanying resources. Fig. 3 illustrates the wireframes that served as the basis for our design of the AI Toolkit v1, which is presented in the next chapter.

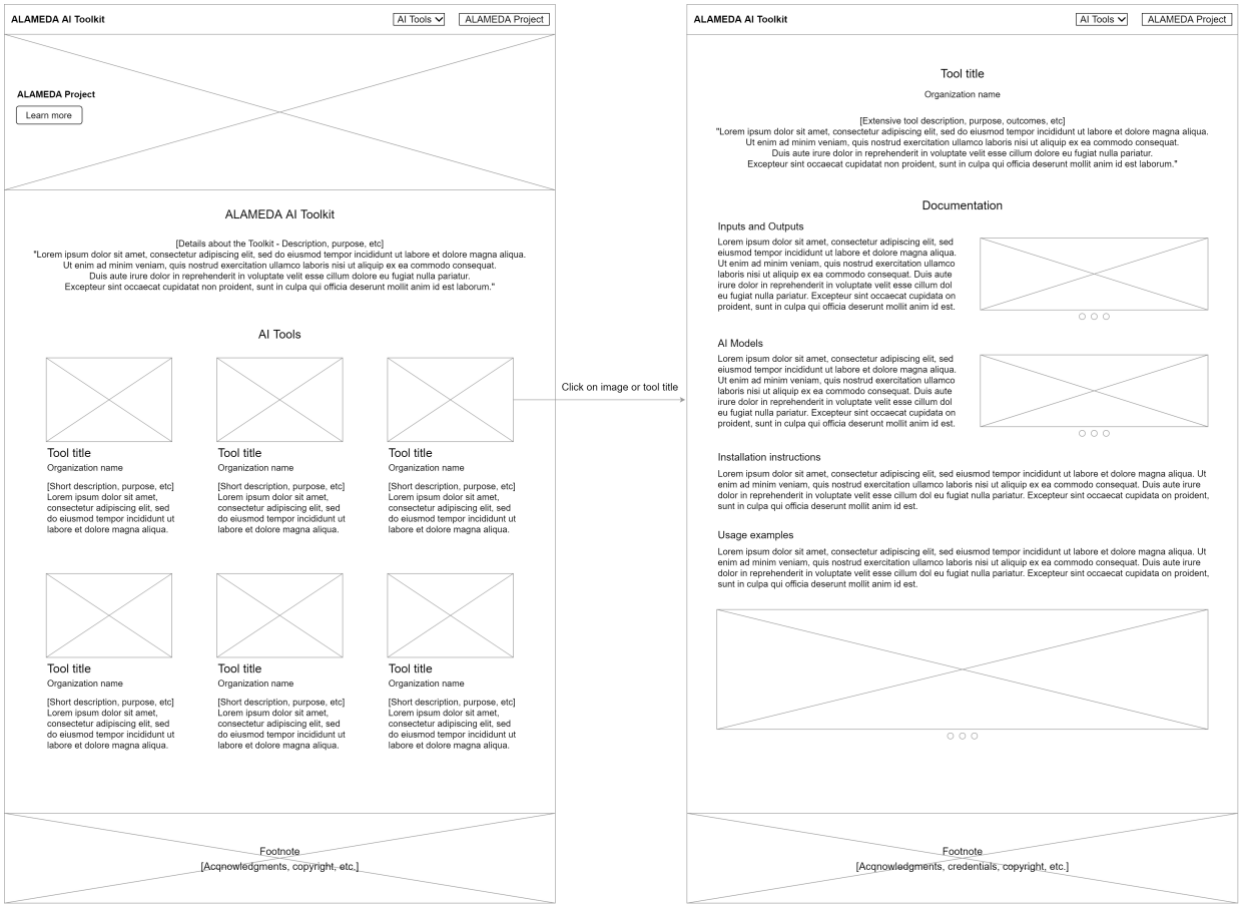

Fig. 3. Wireframes illustrating the AI Toolkit welcome page (left) and the module pages (right).

In the dedicated sub-page of each module, the OpenAPI specifications (see Milestone #1 above) are also included as HTML pages, generated via the OpenAPI-to-HTML generator[2]. For illustration purposes, Fig. 4 displays the API specifications for UNIC's CSAT module, while the rest of the modules have similar pages. In essence, the draft API specification for CSAT from Fig. 2 was converted to the OpenAPI specification in Fig. 4. As seen in the figure, the specification includes a separate section per API endpoint, each of which includes relevant descriptions, sample requests and responses, as well as sample code to submit the request in various popular programming languages.

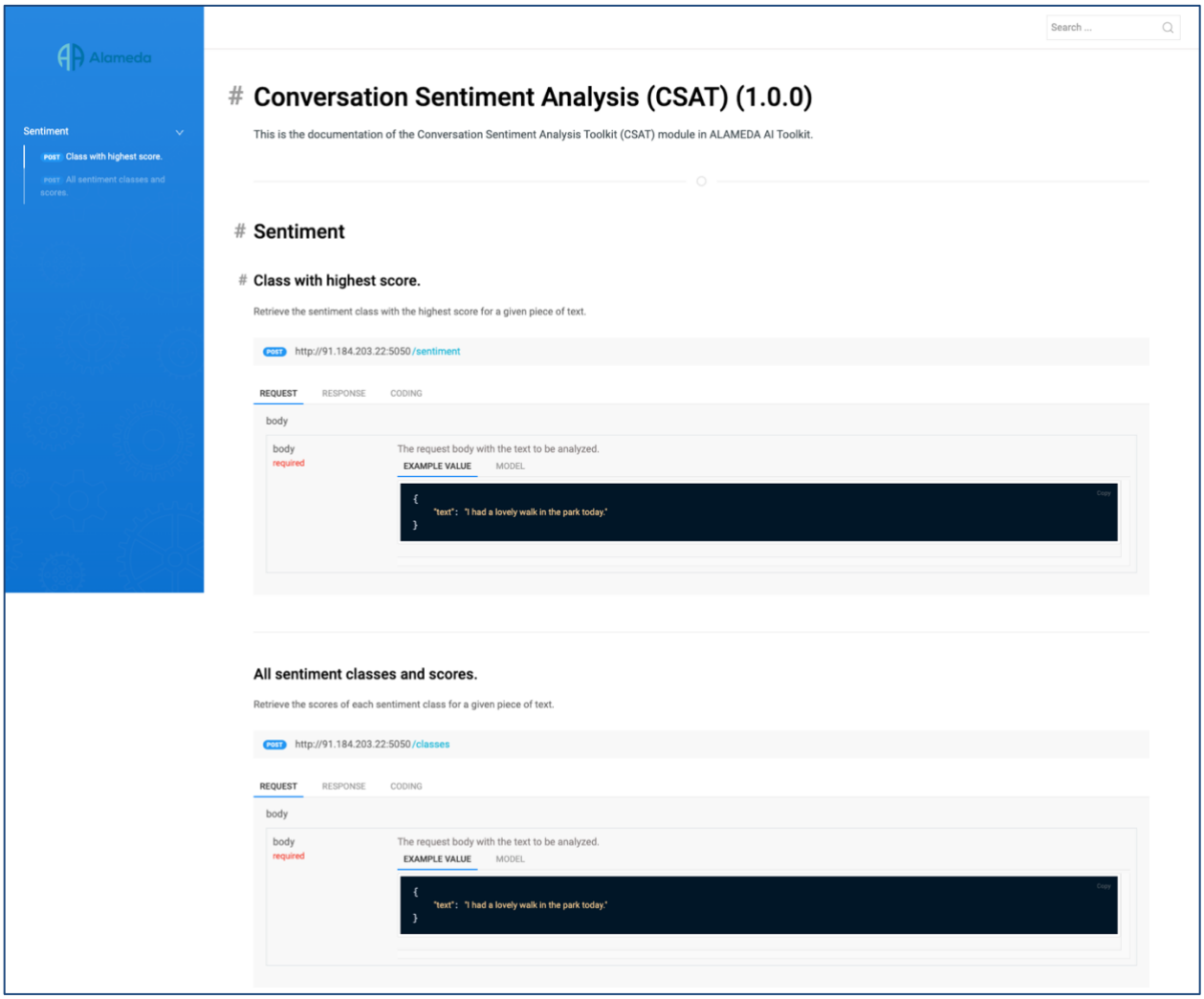

Fig. 4. API specification for UNIC's CSAT module as HTML.

[2] https://asfand-dev.github.io/api-html/

2.3 Compliance to Functional Requirements

The following table assesses the compliance of the AI Toolkit v1 with the functional requirements specified in D5.1 (Table 2, pp. 35-36).

Table 1. Assessment of the AI Toolkit's compliance to the specified functional requirements.

ID	Description of Functional Requirement	Assessment
FUNC-DEP-01	In order to encourage adoption by the community, the AI Toolkit should depend mostly on open-source and freely available libraries instead of commercially available tools that are plagued by licensing complexities.	**SATISFIED** – All the modules in the AI Toolkit v1 depend on open-source and/or freely available libraries and not on commercially available tools.
FUNC-DEP-02	The AI Toolkit will be available as a cloud service after the project, offered via the AIH (WP7).	**NOT SATISFIED YET** – This requirement is aimed for the final iteration of the toolkit.
FUNC-DEP-03	Wherever appropriate, a module should be available for local on-premises installation.	**SATISFIED** – On-premises deployment is possible using the containerized versions (docker images) of the modules provided via the AI Toolkit.
FUNC-PRI-01	Compliance with GDPR must be preserved for all modules in the AI Toolkit.	**SATISFIED** – None of the modules handles sensitive personal information.
FUNC-PRI-02	Data privacy regulations must be respected for all modules in the AI Toolkit.	**SATISFIED** – None of the modules handles sensitive personal information.
FUNC-PRI-03	In the cases of training datasets, these need to be anonymized.	**SATISFIED** – Whenever relevant, training datasets are anonymized.
FUNC-USE-01	The AI Toolkit modules should be readily available "out of the box", without the need to install custom software and/or libraries.	**SATISFIED** – All the AI Toolkit v1 modules are available as docker images that can run anywhere – see also **FUNC-DEP-03**.
FUNC-USE-02	Third parties should be able to experiment with selected real-time data feeds and historic data sources (e.g., patient cohorts) available through the AI Toolkit.	**PARTIALLY SATISFIED** – The respective modules accept historic data but not real-time data feeds, which will be considered for the AI Toolkit v2.
FUNC-USE-03	Third parties should be able to develop new applications using the components available on the AI Toolkit in the AIH.	**NOT SATISFIED YET** – Although the toolkit modules are available as application-agnostic "building blocks" to be integrated in third-party applications, the toolkit was not yet published until now. Thus, there currently do not yet exist such examples of third-party applications utilizing the toolkit's modules.
FUNC-DOC-01	All modules of the toolkit should provide descriptions of their input(s) and output(s), along with the respective data types in the AIH.	**SATISFIED** – All the modules are accompanied by thorough documentation regarding their inputs & outputs – see also Subsection 2.2.1.
FUNC-DOC-02	Wherever appropriate, the AI Toolkit modules should be accompanied by sample datasets, examples, and rich documentation, aiming at a smoother learning curve and end-user adoption.	**SATISFIED** – All the modules are accompanied by thorough documentation, sample datasets, examples, and various other resources.

2.4 Chapter Summary

This chapter presented the approach and methodology behind the design and development of the first iteration of the AI Toolkit, along with an assessment of the level of compliance with the specified functional requirements (D5.1). The next chapter will present the actual implementation of the AI Toolkit v1, focusing on the modules that are currently encompassed.

3. Demonstration of the AI Toolkit v1

This chapter presents in detail the first version of the AI Toolkit and the modules encompassing it.

3.1 Welcome Page

According to the design discussed in the previous chapter, the user is first taken to the toolkit's welcome page, where all the modules currently residing in the toolkit are displayed, along with a brief one-sentence description (see Fig. 5).

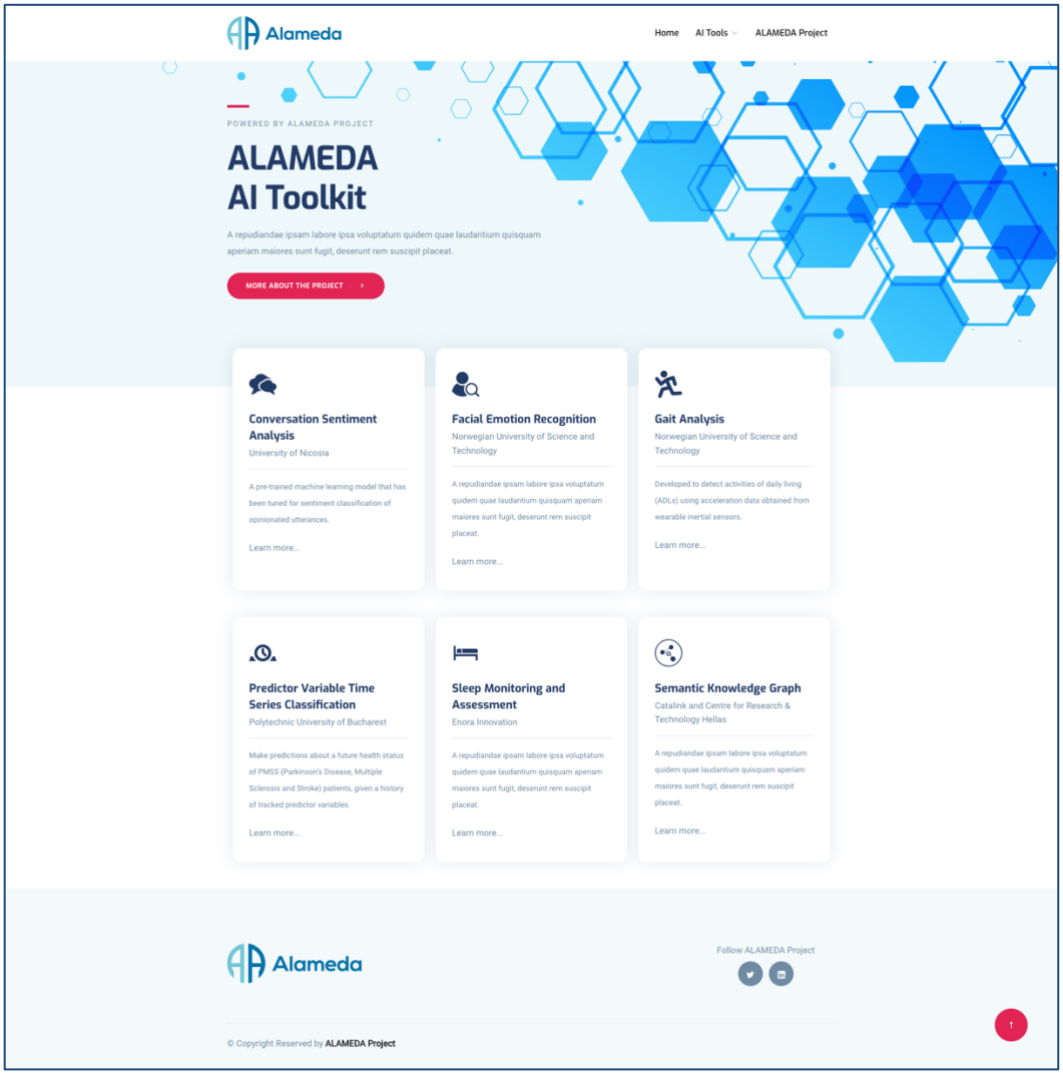

Fig. 5. AI Toolkit v1 welcome page.

By clicking on one of the modules, the user is then navigated to the respective sub-page, as described in the following section.

3.2 AI Toolkit v1 Modules

3.2.1 Conversation Sentiment Analysis Toolkit (CSAT) – Partner: UNIC

The Conversation Sentiment Analysis Toolkit is a pre-trained machine learning model that has been tuned for sentiment classification of opinionated utterances. Currently, the method supports a vocabulary of 183k tokens in two languages, Greek and English, with the prospect to support many more if needed.

Inputs and Outputs

The toolkit receives and processes a sequence of spoken terms (i.e., a user's utterance) up to the length of 150 terms per utterance. It outputs a classification score for each of the three sentiment classes, the positive, the neutral, and the negative class, in a form suitable for the ALAMEDA project.

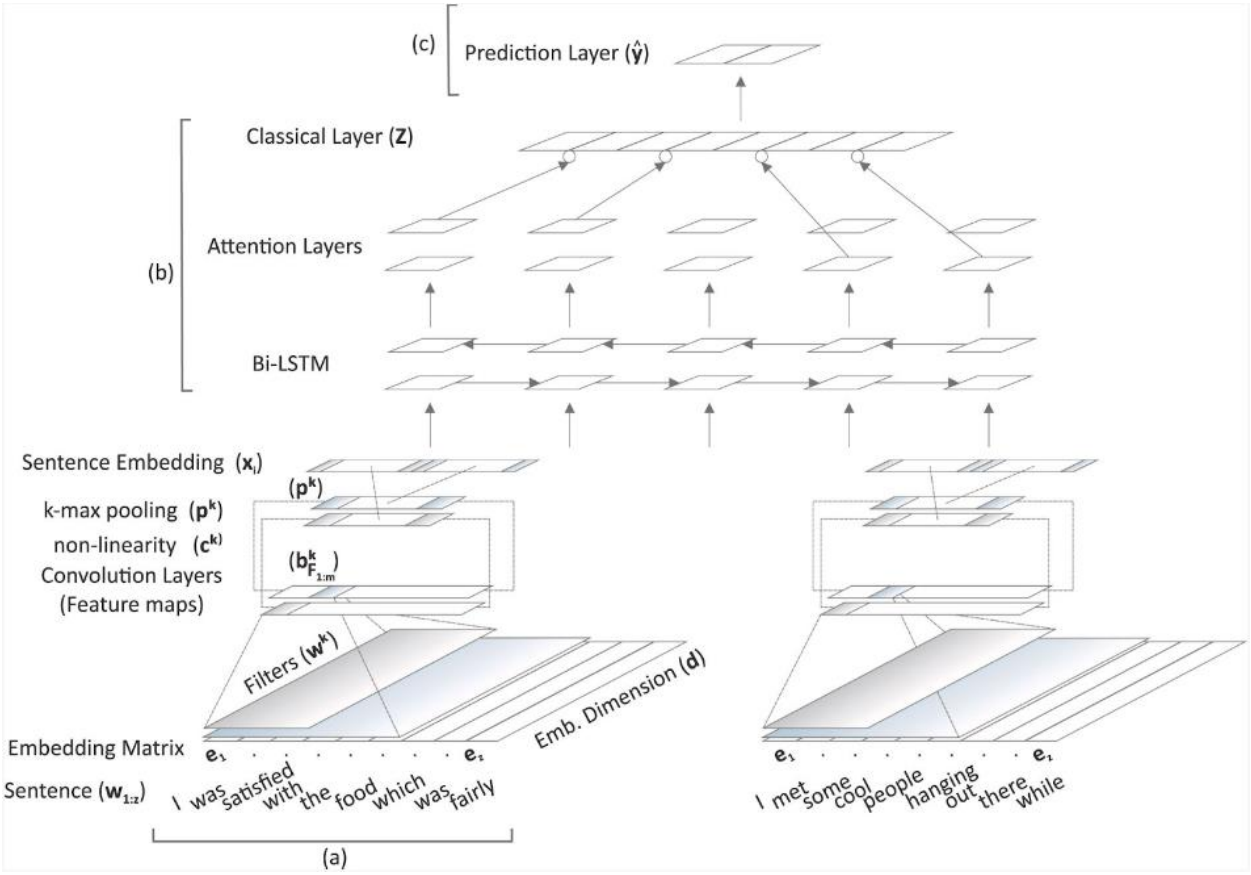

Fig. 6. HolC architecture overview.

Background

HolC[3] is a deep learning model that employs and advances state of the art modules for text and sentiment analysis. It is a feed forward model that consists of three blocks. The first is a Convolutional network that

[3] Agathangelou, P., & Katakis, I. (2022). Balancing between holistic and cumulative sentiment classification. Online Social Networks and Media, 29, 100199.

receives word embeddings and extracts sentence embeddings. The second is a Bi-directional Long Short Memory network (Bi-LSTM) that receives sentences embeddings and outputs unregularized predictions. Then an attention layer augments, filters and transfers neural features into a single layer perceptron which gathers the unregularized predictions and outputs the final prediction. The HolC architecture is illustrated in **Error! Reference source not found.**. HolC provides accurate results outperforming other known architectures of Neural Networks like Convolutional Neural Networks and LSTM networks.

Installation Instructions

The Conversation Sentiment Analysis Toolkit has been developed following industry standards. In this sense, the toolkit has been packaged and uploaded in a dockerized container. It requires the following commands to open ports and run the model:

Step	Command
Open port	`sudo ufw allow 5050`
Login to container registry using guest account	`docker login gitlab.telecom.ntua.gr:5050 -u alameda_ai_toolkit_registry_guest -p ByeYyNesUxqQGs91FzzW`
Pull and run the docker image	`docker run -p 5050:5052 gitlab.telecom.ntua.gr:5050/alameda/alameda-source-code/ai-toolkit/ai-toolkit-registry/csat`
Logout from registry	`docker logout gitlab.telecom.ntua.gr:5050`

Datasets & Samples

The dataset that was used for training and finetuning the Conversation Sentiment Analysis Toolkit included samples from several domains from Amazon's review data[4]. Below are some samples:

1. *I purchased this item as a gift to my daughter. The only complaint she had was the size. She previously had one from 1991 or there about, and it was much smaller and easier to carry in her purse. Other than that, she is very happy with it*
2. *My dogs love chew treats such as rawhide bones and bully sticks. How great, I thought it would be, to buy a dog treat that had some staying power. Staying power it has. What it lacks is any attraction for my dogs. It was a waste of money. Not that much money, but a waste all the same*
3. *The book held my interest, but at times I was frustrated that they kept dwelling on the loss of the twin daughter. Maybe I am a realist, but going on about something that is in the past isn't healthy and just isn't me.*
4. *The toy was ok, my son played with it for a short time. The voice changer aspect is a lot of fun but I don't think the toy will hold older children's attention for long.*

[4] https://jmcauley.ucsd.edu/data/amazon/

5. *Just got back from the vet yesterday. My dog has Lyme disease. So much for front line. We have called in past, but they do nothing. I wonder what other options there are, because this stuff is EXPENSIVE.*

3.2.2 Facial Emotion Recognition (FER) Toolkit – Partner: NTNU

Facial expression (FE) is the most natural and convincing source to communicate human emotions, providing valuable insides to the observer while assessing the emotional incongruities. In health care, the FE of the patient (specifically of neurological disorders (NDs) such as Parkinson's, Stroke, and Alzheimer's) can assist the medical doctor in evaluating the physical condition of a patient, such as fatigue, pain, and sadness. ND patients are usually going through proper observation and clinical tests, which are invasive, expensive, and time-consuming.

In the proposed FER module, a vision transformer (ViT) based FEs recognition framework is developed to classify the facial expression of the user (patients) with an optimal accuracy. Initially, raw images of FEs are acquired from publicly available datasets according to the patient's most common expressions, such as normal, happy, sad, and anger. The framework cropped images through a face detector, extract high-level facial features and fed them to the dense layers for classification. The trained model can be exported through dockerization to other environments and can be evaluated for real-time performance. The FER module is evaluated both qualitatively and quantitatively over standard datasets, i.e., Karolinska directed emotional faces (KDEF), FER 2013 and CK++, etc, showing promising results.

Inputs and Outputs

The trained FER model takes an image in grayscale format and returns the scores (of type float) of each expression class per input image in JSON format. The class with the highest probability labels the final expression of the input image.

Background

In the development of the facial expression recognition (FER) module, various machine learning (ML) and deep learning (DL) algorithms were evaluated over a range of publicly available datasets. In traditional ML techniques, such as random forest, support vector machine, and decision tree, etc., the performance of the random forest technique was satisfactory, but during testing the random forest did not achieve the required level of accuracy. To cope with this challenge, the range of experimentation was extended to DL consisting of sequence (CNN-LSTM, DeepConvLSTM, Video transformer, 3DCNN, etc.) and frame-based (2DCNN, pre-trained models, vision transformer, etc.) learning approaches.

The performance of any DL based technique directly depends on properly annotated datasets, while dataset acquisition is a very challenging task in research. A total of twenty-three datasets were highlighted from the literature, but only seven were acquired and are publicly available.

To cover DL-based sequence learning techniques, we pre-processed and organized the RAVDESS dataset to be incorporated by sequence learning approaches for training and testing. In processing different techniques, both cascade-based and MTCNN face detectors were used to detect and crop the face area,

removing the unessential background from the input frames. While evaluating the DL based approaches, we found that the fine-tuned vision transformer (ViT) model which uses a special attention mechanism for parameter learning outperformed over other DL models. We currently predict four emotions (i.e., angry, happy, sad, and neutral) through our ViT-based trained model. In the future, the ViT-based based model will be extended to predict six or seven facial expressions while conducting training and testing over a combination of different publicly available datasets.

Installation Instructions

The FER toolkit has been packaged in a dockerized container. Once the docker image is hosted on the project's docker registry, it can be accessible by running the following commands:

Step	Command
Open port	`sudo ufw allow 5050`
Login to container registry using guest account	`docker login gitlab.telecom.ntua.gr:5050 -u alameda_ai_toolkit_registry_guest -p ByeYyNesUxqQGs91FzzW`
Pull and run the docker image	`docker run -p 8080:8080 gitlab.telecom.ntua.gr:5050/alameda/alameda-source-code/ai-toolkit/ai-toolkit-registry/fer`
Logout from registry	`docker logout gitlab.telecom.ntua.gr:5050`

For accessing the output JSON files that are in the docker container, the following commands should be run:

1. `docker ps` (this will show the container id)

2. `docker exec -it ""container id"" bash` (in our case `docker exec -it 6d3dc93e5fff bash` – this will convert the given cmd to linux command, by simply typing "" ls "", it will list all the files that are in the docker container. After this, type `exit` and it will back to cmd in)

3. `docker cp 'container id':""`directory of server that pushed docker image to docker hub"" and ""host directory where any file in the docker container can be saved"" (`docker cp 6d3dc93e5fff:/app/Fer_Json_file.json ""D:/Fer_project/container file"""`)

Datasets & Samples

For an extensive evaluation of FER, a literature review was conducted to highlight all possible datasets consisting of facial expression recognition and analysis (FER&A). Twenty-three datasets were chosen to be incorporated for the experimentation of FER&A, where we succeeded to acquire only five datasets including FER-2013, CK++, KDEF, JAFEE, and RAVDESS. The first four datasets in this array were already arranged by the providers in universal facial expressions (FEs) classes such as angry, disgust, fearful,

happy, sad, neutral, and surprise. The latter, RAVDESS, was recorded by 24 professional actors at the Psychology Department of the Ryerson University, consisting of 12 male and 12 female actors (21-33 age range) containing 7356 recordings including only audio (song and speech) and video (speech plus stream of frames) of human FEs (calm, happy, sad, angry, and fearful). We selected only the visual part of it and pre-processed according to our problem, arranging each FE video of all actors into dedicated folders (neutral, calm, happy, sad, angry, and fearful) then extracted frames, detected, and cropped faces to remove the possible effect of background contents on the final output. The final pre-processed RAVDESS dataset contains 1978 videos, where each class approximately consists of 350 videos. In addition, the RAVDESS dataset was further pre-processed for frame-based learning, removing the redundant frames. The trained model generates the output in a JSON format containing the probabilities of four classes, where the maximum probability shows the net expression of the user (Patient).

3.2.3 Gait Analysis (GA) Toolkit – Partner: NTNU

The Gait Analysis (GA) toolkit is a machine learning-based module developed for detecting common activities of daily living (ADLs), such as walking, jogging, going upstairs, going downstairs, sitting, and standing. The GA toolkit contains a pre-trained model based on the smartphone acceleration dataset obtained from wearable inertial sensors.

Inputs and Outputs

The trained gait analysis module takes sample readings of raw acceleration sensor signals in CSV or text format. The input data contains three axial acceleration values with timestamps and the subject ID. The output of the prediction consists of gait class scores for each of the six ADLs (walking, jogging, going upstairs, going downstairs, sitting, and standing) in JSON format containing the probability estimation of each ADL. The output of the prediction can also produce the gait class with the highest gait score.

Background

In the development of the GA toolkit, an array of machine learning methods (random forest, support vector machine, decision tree, logistic regression, etc.) and deep learning models (1D CNN, 2D CNN, LSTM, and hybrid approaches) were already trained and compared, among which the random forest is used as the benchmark model capable of detecting dynamic and static activities of ADL with the lowest error rate. Before feeding the data to a machine/deep learning model, the necessary pre-processing tasks (such as data cleaning, normalization, filtering, balancing, etc.) were performed on the raw sensor signals. Then, the fundamental step of data transformation into many short segments was performed where the sensor signals were framed into partially overlapping windows. In the GA module, a window length of 6.4 seconds (128 samples) and an overlap of 1.25 seconds were applied for segmenting the raw sensor dataset. After segmentation, relevant time domain features are extracted from each window and the resulting feature vectors were used to train machine learning models. The final predictive model has been tuned to extract the best possible performance using an exhaustive grid search with cross-validation over a defined hyperparameter space.

Installation Instructions

The GA toolkit has been packaged in a dockerized container. Once the docker image is hosted on the project's docker registry, it can be accessible by running the following command:

Step	Command
Open port	`sudo ufw allow 5050`
Login to container registry using guest account	`docker login gitlab.telecom.ntua.gr:5050 -u alameda_ai_toolkit_registry_guest -p ByeYyNesUxqQGs91FzzW`
Pull and run the docker image	`docker run -p 8080:8080 gitlab.telecom.ntua.gr:5050/alameda/alameda-source-code/ai-toolkit/ai-toolkit-registry/gait_module`
Logout from registry	`docker logout gitlab.telecom.ntua.gr:5050`

Datasets & Samples

Currently, the GA toolkit model is trained using the publicly available WISDM dataset[5]. This dataset has a total of 1098209 samples collected from 36 volunteer subjects as they performed six daily living activities: walking, jogging, upstairs, downstairs, sitting, and standing for a specific period. All the participants were wearing a built-in motion sensor of the smartphone in their front leg pocket during the experiment execution. An accelerometer sensor with a sampling frequency of 20 Hz was used to record each activity and the data collection process was supervised by a dedicated person to ensure the quality of data. The WISDM dataset is an imbalanced dataset where the walking activity takes up the most, reaching 38.6% while standing only accounts for 4.4%.

3.2.4 Predictor Variable Time Series Classification (VarCls) – Partner: UPB

The objective of the Predictor Variable Time Series Classification service (VarCls) is to allow medical practitioners to make predictions about a future health status of PMSS (Parkinson's Disease, Multiple Sclerosis and Stroke) patients, given a history of tracked predictor variables.

The health status of the patient is determined as:

- A threshold-based classification of a standardized medical test (e.g., Hoehn & Yahr[6] or MDS-UPDRS[7] scales for PD; Romberg[8], MRS[9] scales for Stroke; EDSS scale for MS).

[5] https://www.cis.fordham.edu/wisdm/dataset.php
[6] https://www.physio-pedia.com/Hoehn_and_Yahr_Scale
[7] https://www.movementdisorders.org/MDS/MDS-Rating-Scales/MDS-Unified-Parkinsons-Disease-Rating-Scale-MDS-UPDRS.htm
[8] https://www.physio-pedia.com/Romberg_Test
[9] https://www.mdcalc.com/modified-rankin-scale-neurologic-disability

- A medical practitioner defined classification problem where the classes are defined based on an interpretation of the medical practitioner of a set of medical test results which provide the target health status.

The predictor variables are obtained based on a long term (yearlong) observation of the patient, during which data is collected from two main sources:

- *Objective measurements* obtained from wearable devices (e.g., smart watch, accelerometer-based bracelet, an accelerometer-based smart belt, pressure sensor-based insoles)
- *Subjective Patient Reported Outcomes* (*PROs*) obtained from the interactions of the patient with the ALAMEDA Software Applications (the questionnaire service, the conversational agent, the mood estimation application)

The target variables (health status estimates) are obtained by performing standardized clinical evaluations at specific milestones (e.g., every 3 or 6 months - depending on pilot study).

Inputs and Outputs

The input to the VarCls service is a set of time series, where each time series pertains to one predictor variable. The values of the predictor variables are obtained from the following ALAMEDA toolkits:

The Semantic Knowledge Graph which contains knowledge about the results obtained from PROs filled in by patients throughout the observation period.

The Gait Analysis Toolkit which provides information about gait metrics, daily activity metrics and physical rehabilitation exercise detection.

The Facial Emotion Recognition Toolkit which provides mood estimations based on facial expression analysis, whenever the patient is interacting with an ALAMEDA Software Application.

The Conversational Sentiment Analysis Service which provides mood estimations based on text analysis, whenever the patient engages in free-text dialogue with the ALAMEDA Conversational Agent.

The ENORA Sleep Monitoring Service which provides information about sleep metrics based on a combined input from wearable devices (e.g., Fitbit Smartwatch, accelerometer-based bracelet) and smart mattresses (e.g., Withings Sleep Mat).

The exact set of PRO results and toolkit extracted metrics is specific to each monitored disease (PD, MS, and Stroke).

The inputs are provided to the VarCls service as a CSV file, whereby each row contains the values of predictor variables, as well as information about the timestamp of the measurement and the patientID to which the measurement belongs.

The service invocation also requires the specification of an AI model name to use in the prediction (see Section on AI Models).

The output of the VarCls service is a CSV file containing one or more target variables together with the probability distribution for each target variable value.

AI Models

Several AI models underlie the VarCls service. Each AI model is specific to the target variable of a particular neurological disease (PD, MS, or Stroke).

The VarCls service operates with input time series whose variable values are numerical (e.g., number of steps performed during a day, number of hours slept, average walking speed throughout a day), categorical (e.g., estimated mood state) or ordinal (e.g., Likert-scale response on a questionnaire) in nature.

As such, the employed AI models fall under two main categories:

- **Aggregative feature extraction**: this class of models will make use of statistical feature extraction over time series windows, to which ensemble models such as RandomForest or XGBoost classifiers are applied.
- **Express time series classification**: make use of well-known time series classification models (e.g., BOSSEnsemble, Catch22Classifier, SupervisedTimeSeriesForest, TSFreshClassifier, HIVE COTE v2) to directly classify the time series of predictor variables.

Installation Instructions

The VarCls toolkit has been packaged in a dockerized container. Once the docker image is hosted on the project's docker registry, it can be accessible by running the following command:

Step	Command
Open port	`sudo ufw allow 5050`
Login to container registry using guest account	`docker login gitlab.telecom.ntua.gr:5050 -u alameda_ai_toolkit_registry_guest -p ByeYyNesUxqQGs91FzzW`
Pull and run the docker image	`docker run -p 8080:8080 gitlab.telecom.ntua.gr:5050/alameda/alameda-source-code/ai-toolkit/ai-toolkit-registry/var_cls`
Logout from registry	`docker logout gitlab.telecom.ntua.gr:5050`

3.2.5 Semantic Knowledge Graph (SemKG) – Partners: CERTH & CTL

In order to prevent interested stakeholders from writing complex SPARQL queries for accessing the knowledge residing in the ALAMEDA ontology, SemKG serves as the sole entry point for inserting and retrieving data.

The SemKG component encompasses:

- The semantic model, i.e., the ALAMEDA ontology.
- Instance data fed to the ontology as outputs from other ALAMEDA components.
- The repository for persisting the populated semantic model, i.e., RDF triplestore.
- The REST API for interacting with the information residing in the repository.

Inputs and Outputs

Through its RESTful API, SemKG allows submitting HTTP requests: POST requests for ingesting component outputs into the ontology; GET requests for retrieving information from the ontology.

However, due to the sensitive nature of personal information stored in the ontology, the SemKG module in the AI Toolkit gives access only to anonymized and aggregate information as well as schema knowledge (i.e., ontology concepts and attributes).

For the first version of the toolkit, only schema information is available, with regards to sleep disorders and sentiment assessment, which are retrieved in JSON-LD format, in order to comply to the W3C semantic interoperability standards. Our motivation is for third-party researchers to adopt the schemes in their applications.

Background

SemKG is fully compliant to W3C standards: RDF/OWL for semantically representing all pertinent concepts, SPARQL for querying the semantic Knowledge Graph, JSON-LD for representing the retrieved information in a format facilitating semantic interoperability.

Installation Instructions

The Semantic Knowledge Graph module requires a SPARQL-enabled RDF triplestore, which is defined by its SPARQL endpoint URL. To pull and run the SemKG Docker image use the following commands:

Step	Command
Login to container registry using guest account	`docker login gitlab.telecom.ntua.gr:5050 -u alameda_ai_toolkit_registry_guest -p ByeYyNesUxqQGs91FzzW`
Pull and run the docker image	`docker run -p 4567:4567 -e triplestore_endpoint="YOUR_ENDPOINT_URL" gitlab.telecom.ntua.gr:5050/alameda/alameda-source-code/ai-toolkit/ai-toolkit-registry/semkg`
Logout from container registry	`docker logout gitlab.telecom.ntua.gr:5050`
The SemKG API is accessible at	`http://0.0.0.0:4567`

Datasets & Samples

JSON-LD file representing the sentiment assessment scheme:

```
[
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "scheme": "sentimentScheme",
    "@id": "Positive"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "scheme": "sentimentScheme",
    "@id": "Neutral"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "scheme": "sentimentScheme",
    "@id": "Negative"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "scheme": "sentimentScheme",
    "@id": "Other"
  },
  {
```

```
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "broader": "Other",
    "scheme": "sentimentScheme",
    "@id": "Angry"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "broader": "Other",
    "scheme": "sentimentScheme",
    "@id": "Disgust"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "broader": "Other",
    "scheme": "sentimentScheme",
    "@id": "Fear"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "broader": "Other",
    "scheme": "sentimentScheme",
    "@id": "Surprise"
  }
]
```

JSON-LD file representing the Stroke-related sleep disorders scheme:


```
[
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state deep sleep (in seconds).",
    "scheme": "strokeSleepDisordersScheme",
    "@id": "deepSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state light sleep (in seconds).",
    "scheme": "strokeSleepDisordersScheme",
    "@id": "lightSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state REM sleep (in seconds).",
    "scheme": "strokeSleepDisordersScheme",
    "@id": "remSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Ratio of the total sleep time over the time spent in bed.",
    "scheme": "strokeSleepDisordersScheme",
    "@id": "sleepEfficiency"
```

```
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "The Sleep score is a very simple and intuitive way to
understand how well you slept. It measures every night's sleep and provides a
score out of 100 points based on 4 key inputs:\n\nDuration (total time spent
sleeping) Depth (part of night spent in restorative phases and deep sleep)
Regularity (consistency between your bed- and rise-times) Interruptions (time
spent awake)\nSleep duration and depth are the most important factors for
raising sleep score. The key factors in the sleep score are sleep regularity
and sleep hygiene. An improvement in these hygiene measures will improve the
overall sleep experience.",
    "scheme": "strokeSleepDisordersScheme",
    "@id": "sleepScore"
  }
]
```


JSON-LD file representing the Parkinson's-related sleep disorders scheme:


```
[
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Wellness metric, available for all Sleep and Sleep Analyzer
devices",
    "scheme": "pdSleepDisordersScheme",
    "@id": "breathingDisturbancesIntensity"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state deep sleep (in seconds).",
    "scheme": "pdSleepDisordersScheme",
```

```
    "@id": "deepSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Maximum heartrate during the sleeping session.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "hrMax"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Minimum heartrate during the sleeping session.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "hrMin"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state light sleep (in seconds).",
    "scheme": "pdSleepDisordersScheme",
    "@id": "lightSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Level of LUX inside the room",
    "scheme": "pdSleepDisordersScheme",
    "@id": "luminocity"
  },
```

```
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Count of the REM sleep phases.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "nbRemEpisodes"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Time windows during which the person got out of bed.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "outOfBed"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Number of times the user got out of bed during the night.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "outOfBedCount"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Time windows and position of sleep, e.g., straight,
crunch.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "positions"
  },
  {
```

```
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Duration in state REM sleep (in seconds).",
    "scheme": "pdSleepDisordersScheme",
    "@id": "remSleepDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Average respiration rate.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "rrAverage"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Maximal respiration rate.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "rrMax"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Minimal respiration rate.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "rrMin"
  },
  {
    "@type": "skos:Concept",
    "@context": {
```

```
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Ratio of the total sleep time over the time spent in bed.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "sleepEfficiency"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "The Sleep score is a very simple and intuitive way to
understand how well you slept. It measures every night's sleep and provides a
score out of 100 points based on 4 key inputs:\n\nDuration (total time spent
sleeping) Depth (part of night spent in restorative phases and deep sleep)
Regularity (consistency between your bed- and rise-times) Interruptions (time
spent awake)\nSleep duration and depth are the most important factors for
raising sleep score. The key factors in the sleep score are sleep regularity
and sleep hygiene. An improvement in these hygiene measures will improve the
overall sleep experience.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "sleepScore"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Total snoring time",
    "scheme": "pdSleepDisordersScheme",
    "@id": "snoring"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Numbers of snoring episodes of at least one minute",
```

```
    "scheme": "pdSleepDisordersScheme",
    "@id": "snoringEpisodeCount"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Time windows during which the person did not move.",
    "scheme": "pdSleepDisordersScheme",
    "@id": "still"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Room temperature",
    "scheme": "pdSleepDisordersScheme",
    "@id": "temperature"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Time spent awake (in seconds).Time spent awake (in
seconds).",
    "scheme": "pdSleepDisordersScheme",
    "@id": "wakeUpDuration"
  },
  {
    "@type": "skos:Concept",
    "@context": {
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "scheme": "skos:inScheme",
      "definition": "skos:definition",
      "broader": "skos:broader"
    },
    "definition": "Time spent awake in bed after falling asleep for the 1st
time during the night",
```

```
    "scheme": "pdSleepDisordersScheme",
    "@id": "waso"
  }
]
```

3.2.6 Sleep Monitoring and Assessment – Partner: ENO

The ENORA Sleep Analysis Toolkit is designed together with PD clinicians and is focusing on the following services:

Tool 1 – Position Classification: The tool classifies the grid data of the prototype ENORA pressure sensing mattress top to identify the specific category of PD status regarding the position of the patient on the bed.

Tool 2 – Sleeping Patterns Analysis: This analysis receives as inputs a time series data and aims to detect known PD patterns. The position identification performed in the Position Classification tool is coupled with continuous data collected from the Withing sleep analysis sensor in order to reveal further sleep insights. This tool can be used in combination with the Position Classification tool or independently, exploiting the use of sleep analysis data specifically adapted to PD disorders. Overall, this tool is combining: (a) Sleeping positions over night (Tool 1); (b) Continuous heart rate; (c) Snore detection; (d) Breathing patterns and breath disturbances; (e) Sleep cycles; (f) Sleep score. The assessment is based on the identification of specific PD disorders and classifies the PD patient's medical condition with respect to PD sleep quality (with a level of uncertainty).

Inputs and Outputs

Input:

- ENORA mattress top pressure sensor data series (Tool 1).
- Withings under mattress Sleep Sensor data series (Tool 2). Will require sensor authentication to use the data in ALAMEDA.

Output:

- PD sleep quality and medical condition assessment. This is work in progress with medical partners because no relevant standard exists.

Background

Parkinson's Disease is a complex movement disorder that is affecting millions of people around the world. It's estimated that two-thirds of those afflicted with Parkinson's Disease struggle to get quality sleep. In addition, sleep problems are increasingly recognized as a potential early indicator of Parkinson's Disease.

ALAMEDA is directly addressing sleep quality for PD and the Sleep Monitoring & Assessment Toolkit for PD will provide assessment of the PD sleep quality and of their medical condition based on continuously collected data series for extended periods using non-intrusive/wear-nothing technologies. The use of a commercial device that is used globally for sleep analysis will improve the outreach of ALAMEDA and will

enable the use of the produced AI sleep analysis algorithms by users worldwide, via the ALAMEDA Innovation Hub.

Installation Instructions

The Sleep Monitoring Toolkit is packed in a dockerized container. Internally it starts a web browser listening to port 80. In order to run the container simply run:

From a command line or docker desktop: `docker run -d -p 80:80 enorawebapi`

From a web browser view the swagger view and test `http://localhost/swagger/index.html`

Datasets & Samples

Currently the Sleep Monitoring Toolkit accepts JSON objects (key value pairs) of the sleep mattress and Withings sensor raw data. It returns position, out of bed, stillness. Data are pushed at 20 seconds interval (sleep mattress) and 60 seconds of the Withings (heart rate and respiration rate). This is a prototype version, and the models are being upgraded with added modalities.

3.3 Online Demo

The first version of the toolkit is available at https://alameda.catalink.eu/

3.4 Chapter Summary

This chapter presented in detail the AI Toolkit, currently consisting of six AI-powered modules. The design of the toolkit is simple, with a landing/welcome page and separate sub-pages for each module. For each module, rich descriptions are provided, along with installation and execution instructions, guidelines, background information and relevant/accompanying resources. The next chapter focuses on the approach and guidelines towards implementing the ALAMEDA AI Toolkit and the overall ALAMEDA platform integrating the individual ALAMEDA components and applications.

4. ALAMEDA Platform Integration

The ALAMEDA consortium aims to design and implement the novel platform that will provide sophisticated and valuable features and functionalities to PMSS (Parkinson's, Multiple Sclerosis, and Stroke) patients. To this end, the design specifications of the ALAMEDA platform incorporate and exploit multiple open-source technologies, frameworks and tools which are combined in order to implement the multi-tier ALAMEDA architecture, as documented in D5.1. In particular, the ALAMEDA architecture is composed of multiple components, which should be effectively and smoothly integrated to provide the designated rich set of functionalities and features of the ALAMEDA platform.

The development of complex software products requires the adoption of a flexible and solid development process. Within the context of ALAMEDA, the adopted development process, as documented in D5.1, has two basic axes. For the main development, which includes the basic infrastructure and the core of the AI Toolkit, a two-step iterative process is followed where two sets of specifications (on M12 and M20, with deliverables D5.1 and D5.4, respectively) and two software bundles that implement these two core elements of the ALAMEDA platform based on these specifications (on M18 and M30, with the current deliverable and D5.5, respectively) are planned and delivered. The rest of the software components and applications follow a spiral methodology with multiple iterations which produce versions of the components in order to be effectively integrated with the basic infrastructure and the core of the AI Toolkit.

The main goal of the adopted development process is to produce incremental versions of the ALAMEDA platform, in which the various incremental versions of the components are effectively and smoothly integrated, verified and validated. Each produced incremental version of the ALAMEDA platform constitutes an integration cycle of the development phase of the ALAMEDA platform and is provided as the basis for the next iteration. To meet its goals, the development process is supported by a solid and robust integration process that clearly defines the performed steps during the implementation and integration activities and guarantees the quality of the implemented software artefacts.

As briefly described in D5.1 and illustrated in Fig. 7, the ALAMEDA integration process is based on the following pillars:

- **A set of mature and well-established open-source integration tools and techniques** that facilitate the planning, design, and collaboration of the involved development teams.
- **A solid integration strategy** that enables the planning, design, and delivery of the various components, with the help of the aforementioned integration tools and techniques.
- **A robust integration plan** that defines the integration cycles of the ALAMEDA platform and the integration activities that will be performed during each integration cycle.
- **A solid integration testing process** that defines the validation and verification activities that will be performed on each integration cycle.

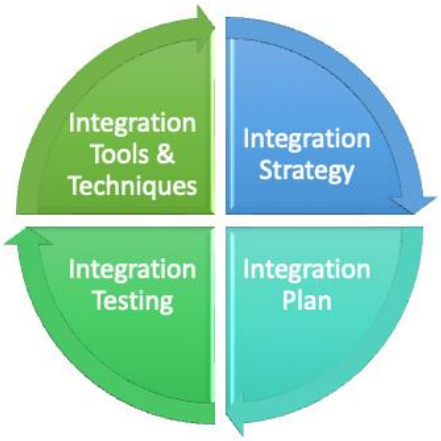

Fig. 7. ALAMEDA platform integration process pillars.

The following sections present in detail each of the main pillars of the ALAMEDA integration process.

4.1 Integration Tools and Techniques

The integration process of any software product can be rather complex and challenging. For this reason, the process is backed up by a set of carefully selected tools and techniques that facilitate the proper planning, design and collaboration of the various development teams which are engaged in this process. These tools are selected based on their establishment and wide adoption in the development process as well as their licence as open-source libraries and frameworks are preferred. In addition, several other aspects are taken into consideration during the selection process. The suitability for the designed process and the level of maturity in terms of features constitute the main criteria, as the ultimate goal is the successful and smooth execution of the integration process, as well as the utilisation of their features which enable the monitoring and management of the process during the whole execution.

To this end, the list of integration tools and techniques includes the following:

- **Source code versioning and management tools** are fundamental tools for the effective integration process execution. In particular, the cornerstone of each developed software artefact is considered the effective source code versioning and management. In the industry, Git is considered the de-facto solution as it is the well-established software solution which offers a version control system with a large list of features and functionalities currently adopted by all software development projects. It enables flexibility, security, and performance and more importantly the collaboration of multiple parties on the same source code with increased trust and monitoring capabilities. Within the context of ALAMEDA, the technical team decided to utilize the online Git repository **Gitlab** as the source code version and management tool of ALAMEDA.
- **Automated building tools** enable the compilation and building process of the produced software artefacts depending on the utilized programming language. The most important tools of this category are the building and containerization tools that facilitate the compilation and building of the implemented components in a way that they are easily deployed on the underlying infrastructure in the most effortless and transparent manner. Within the context of ALAMEDA,

the well-established **Maven and Gradle** tools are selected for the automated building of the Java-based components, while the de-facto framework **Docker** is utilised as the containerization tool of the project.

- **Unit and Integration testing frameworks** constitute the main tools utilized in order to safeguard the software quality as they facilitate the design, implementation and execution of unit testing suites per component as well as integration suites for the platform features where multiple components are involved. These frameworks are dependent on the programming language utilized for the implementation of the components hence multiple frameworks will be exploited such as Junit, TestNG, unittest, Jasmin and Mocha.
- **Issue tracking tools** provide the means to perform the planning and the monitoring of the progress of the whole development and integration process. As such, these tools enable the planning, design, and implementation of new features as well as the tracking of defects and bugs during the implementation and integration phase. Within the context of ALAMEDA, **Gitlab** is also selected for issue tracking as it provides the most effective and efficient issue tracking tool.
- **Continuous Integration and Continuous Delivery (CI/CD) tools** enable the integration and automatic deployment of the produced artefacts on the appointed infrastructure during the integration cycles. The purpose of this process is to perform the system-wide integration testing with several automated and/or manual tests. Within the context of ALAMEDA, **Gitlab Runner** is the preferred choice for the CI/CD process in order to enable the design and execution of various CI/CD pipelines depending on the involved software artefacts.
- **Source code quality testing tools** facilitate the early defect identification and formal verification of the quality of the produced software artefacts. Hence, the quality, reliability, security, and maintainability of the source code is increased by identifying and eliminating the error-prone parts, the vulnerabilities and security threats of the source code. Within the context of ALAMEDA, **SonarQube**, the well-established open-source solution, is selected as the code analysis and quality assurance tool.

4.2 Integration Strategy

The main pillar of any integration process where multiple components are designed by different development teams and later integrated into one fully functional software platform is the definition of a solid and concrete integration strategy. The integration should contain the complete steps that should be performed as well as their order of execution. On each integration iteration, the steps are executed in order to produce the incremental version of the platform. Within the context of ALAMEDA, the formulated integration strategy includes the following:

a) The concrete steps that should be executed during the production of the incremental version of the platform.
b) The concrete steps that should be executed during the implementation of an incremental version of each component of the platform.
c) The concrete steps that should be executed during the integration of the produced incremental versions of the components towards the production of the incremental version of the platform.

Besides providing the guidelines for the complete integration activities, the formulated integration strategy also provides the required input for the formulation of an integration plan that will be followed during the whole development phase of the platform. The integration plan will guarantee the on time and effective delivery of the incremental versions of the platform with the highest software quality.

For each incremental version of the platform, it is imperative that a set of steps is performed in order to facilitate but also to safeguard the process. The list of steps (see also Fig. 8) includes:

a) **The identification of the components** which are involved in the specific iteration and the features that will be introduced by these components on the specific iteration in accordance with their design specifications.
b) **The identification of the dependencies between the involved components** and their prioritization.
c) **The resolution of the identified dependencies** with the appropriate technical interfaces which are exposed by the involved components in order to facilitate their integration and their prioritization.
d) **The definition of the integration plan** for the specific iteration which takes into consideration all the output of the previous steps.

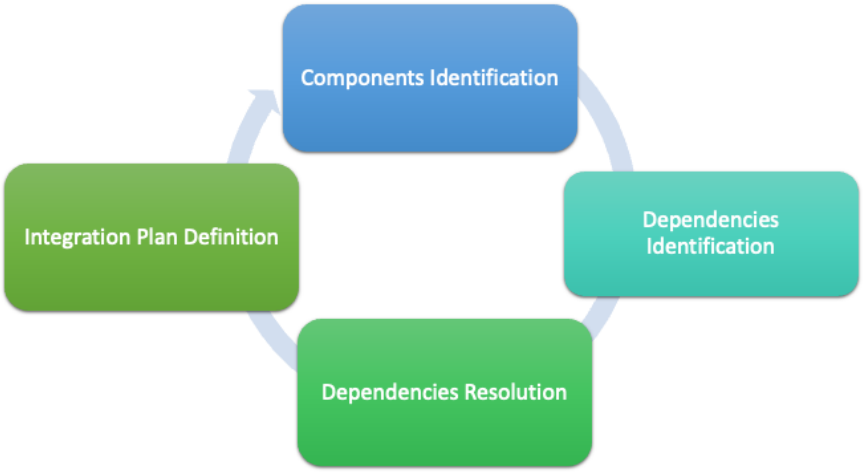

Fig. 8. ALAMEDA platform integration cycle.

The development of each component included in the integration process in an iterative manner includes a list of steps which are supported by the appropriate integration tools and techniques (see also Fig. 9):

a) **Source code definition and commit to the Git repository**: For each component, the responsible development team undertakes the task of compiling the respective source code that delivers the new features based on the plan and design specifications. The outcomes of this work are pushed to the respective Git repository of the component.
b) **Automated build**: The new source code is built utilising the respective automation building tool depending on its implementation language where required.

c) **Automated test**: The produced new version of the component is verified and evaluated based on the designed unit testing suite which is composed by the respective unit testing framework. Provided that the results are acceptable the new version of the component is made available to the next steps and any possible defects identified are inserted into the issue tracking tool.
d) **Containerisation and Deployment**: The successfully verified and evaluated incremental version of the component is containerised and made available for the upcoming integration process.

Fig. 9. ALAMEDA component development cycle.

The integration of all the produced incremental versions of the components towards the production of the new incremental version of the platform is based on the following concrete steps (see also Fig. 10):

a) **Component implementation**: The new incremental version of the component is produced following the steps described for the development of each component above. The development team which is responsible for the development of each component works independently in order to produce this new version of the component.
b) **Component integration**: The newly available version of the component is available and deployed into a new container on the platform's infrastructure. The integration of the produced component involves the execution of the integration suite that involves this component. In cases where issues or defects are identified they are resolved by the respective development teams. If the produced software quality is above the defined threshold, the new version of the component is included in the next step.
c) **Platform integration**: This step performs the system-wide integration of the produced components. It is the last phase of the process and involves all the components that have successfully passed all the previous steps of the process. In this step, the platform is verified and evaluated as a whole, hence the complete integration suite is executed and the respective results are evaluated to verify the new features of the platform. Once all tests have successfully passed, the new incremental version of the platform is released in order to be evaluated by the ALAMEDA stakeholders.

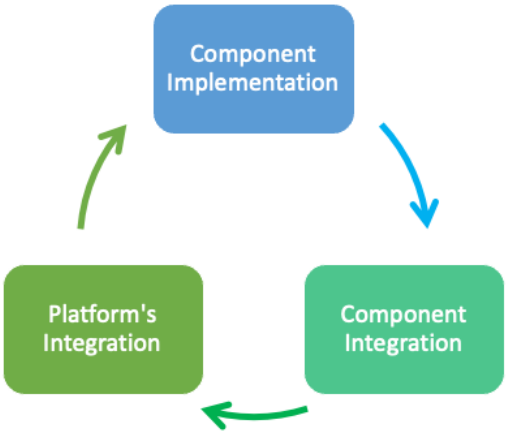

Fig. 10. ALAMEDA platform integration process.

4.3 Integration Plan

As documented in the ALAMEDA Integration Strategy, the formulation of the integration plan receives as input the identification of the components of the integration cycle and the identification of the dependencies between these components. The resolution of these dependencies drives the formulation of the integration plan which includes all the necessary actions that will be performed in the integration cycle in order to guarantee the smooth integration process.

To this end, the technical team identified the dependencies between the various components following a simple and concrete rule:

"A component depends on another component if it utilises at least one interface or service of the other component in order to deliver its functionalities."

Based on this rule, the technical team composed the dependency matrix of the ALAMEDA platform's components which is illustrated in Fig. 11. The matrix illustrates the dependency of all ALAMEDA Digital Companion Apps with the rest of the components and especially with the Semantic Knowledge Graph, while also depicting the dependency of the AI-powered Analytics components with the AI Toolkit which contains the modules described in section 3.2. Finally, all components have a dependency with the User Security component that undertakes the user management, authentication, authorisation and role management functionalities of the ALAMEDA platform.

As the development of the ALAMEDA platform follows a two-step iterative process where two software bundles that implement the basic infrastructure and the core of the AI Toolkit are delivered on M18 with the current deliverable and on M30 with deliverable D5.5, those two milestones constitute the main internal milestones of the platform. It should be noted that the rest of the software components and applications follow a spiral methodology with multiple iterations between these two internal milestones that produce incremental versions of the components. During these two internal milestones, the ALAMEDA Integration Strategy is followed and supported by the ALAMEDA integration tools and techniques and new incremental versions of the components are produced based on their design specifications in order to deliver new features and functionalities. Moreover, each new incremental

version of the components provides a set of adjustments and refinements on the existing features and functionalities in order to better address the user requirements and the feedback that will be received by the ALAMEDA pilots.

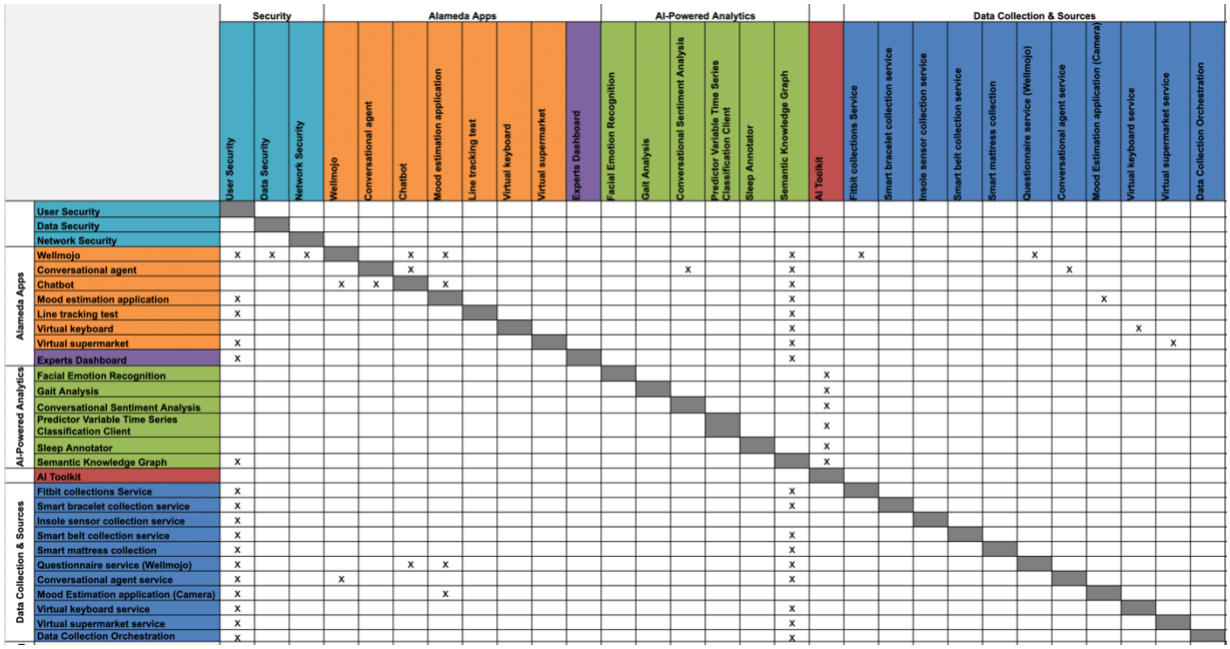

Fig. 11. ALAMEDA dependency matrix.

Fig. 12 depicts the ALAMEDA Integration Plan. As illustrated during the first development phase till M18 all the platform's components produced their first incremental versions. The components constituting the core of the AI Toolkit deliver their first major version, while the rest of the components produce their initial prototype versions and a set of incremental versions towards their production version. It should be noted that the integration of these components is an ongoing process that will be completed till M30 as planned. Additionally, the components not included in the AI Toolkit produce different intermediate internal releases on top of their production version towards the release of their incremental version that contain updates and optimisations that resolve all the issues identified during the integration process of each platform incremental release till the release of their final version. On M20 the second version of the design specifications of the AI Toolkit components are collected which will drive the implementation of the AI Toolkit components' final version. As depicted also in the integration plan, the second and final incremental version of the ALAMEDA platform will incorporate the final releases of all the components that will be effectively integrated in order to formulate the fully functional ALAMEDA platform release.

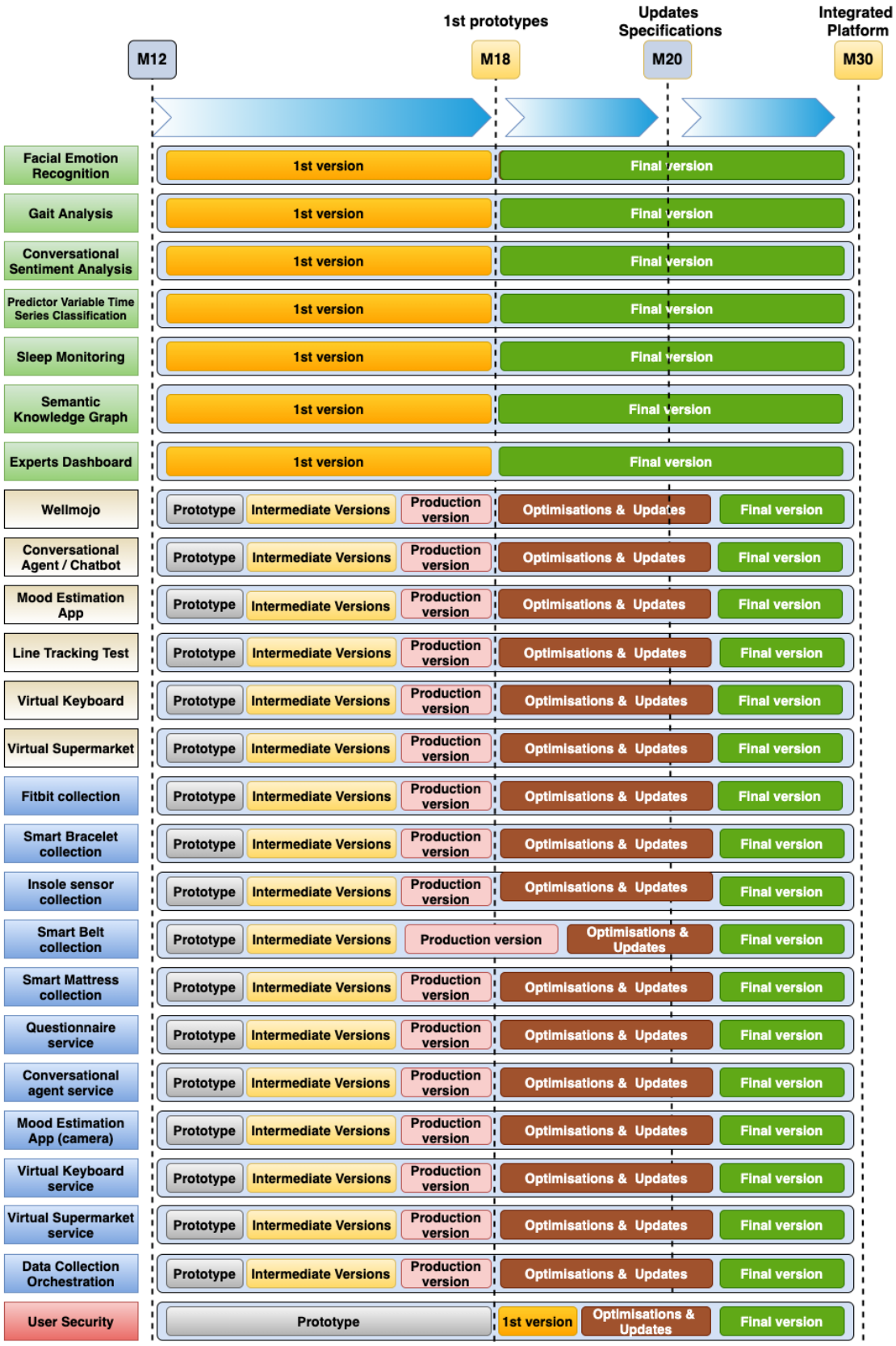

Fig. 12. ALAMEDA integration plan.

4.4 Integration Testing

Software testing constitutes a fundamental aspect of every software development project as it safeguards the software quality of all produced software artefacts as independent modules as well as of the complete integrated software solution that is produced. To this end, integration testing assists the smooth integration process as it undertakes the detection of bugs, defects and problems of the produced solution starting from the development of each separate component till the combination and integration of all components that formulate the designed software solution. Moreover, it provides the required verification that when the multiple components are combined and interconnected offer the designed functionalities and features per the components' design specifications and per the software solution's architecture.

Software testing is divided into two main categories, namely **unit testing** and **integration testing**. These testing activities are designed and performed within a different context and at a different phase of the software development process. In particular, unit testing aims at verifying the software quality and functionalities at a component level, hence it is executed during the implementation phase of each component. To this end, unit testing is performed during the integration cycles when a new incremental version of a component is produced, it is usually automated through unit testing frameworks and verifies that the functionalities and features offered by the new incremental version of the component are working as expected. Unit testing enables the discovery of defects at an early stage by assessing its behaviour thus it optimises the development process in both time and effort. Typically, the defects and bugs that are identified at an early stage through unit testing are usually not complex and easier to fix than the ones identified on an integrated solution.

Within the context of ALAMEDA, as per the ALAMEDA integration strategy each component that is developed will be thoroughly tested with multiple unit tests by the partner who was responsible for the development of the specific component. The execution of the unit tests is performed during the component implementation and before the component integration stage. To facilitate the unit testing process, a common template has been defined in order to document each unit test that will be executed as presented in **Error! Reference source not found.**. In accordance with this template, each unit test is identified through a unique reference ID (Test Case Reference ID) and a unique name (Test Case Name) that facilitate their tracking during their execution. In addition to this, the respective component name, for which the test is designed, is documented. Since unit tests include a number of steps that should be executed, these steps are clearly documented in the "Description of the test case". Finally, all steps typically require some input data that will be fed into the component's functions in order to be validated and verified during each step. Hence, the input data is properly documented in the "Input of the test case". In the same manner, the expected output from each step's execution is documented in the "Output of the test case".

Table 2. ALAMEDA unit test template.

ALAMEDA Unit Test	
Test Case Reference ID	TC#
Test Case Name	TestX
Component Name	ComponentX
Description of the test case	
Complete description of the performed steps and the purpose of the test	
Input of the test case	
Description of the input utilised in the test case	
Output of the test case	
Description of the output expected from the test case execution	

Following the same approach, a common template has been defined in order to document each integration test that will be executed as presented in **Error! Reference source not found.**. As the integration tests will validate a functionality or feature of the platform where multiple components are involved and exchange information, the template is slightly differentiated from the previous template. In detail, each integration test is also identified through a unique reference ID (Test Case Reference ID) and a unique name (Test Case Name) that facilitate their tracking during their execution. As during integration testing multiple components are involved, the "Components involved" lists the names of these components. The number of steps that should be executed are clearly documented in the "Description of the test case". Moreover, the "Input of the test case" documents the required input for each while the "Output of the test case" documents the expected output from the execution of each step.

Table 3. ALAMEDA integration test template.

ALAMEDA Integration Test	
Test Case Reference ID	IT#
Test Case Name	TestX
Components involved	ComponentX, Component Y, Component Z, …
Description of the test case	
Complete description of the performed steps and the purpose of the test	
Input of the test case	
Description of the input utilised in the test case	
Output of the test case	
Description of the output expected from the test case execution	

4.5 Chapter Summary

This chapter presented the ALAMEDA platform integration approach, which is the focus of T5.4. The deliverable is concluded with the next chapter featuring a final discussion on concluding remarks and pointers for the next steps.

5. Conclusion & Next Steps

This document is supplementary to the demonstrator deliverable D5.3 and reports on the first iteration of the AI Toolkit, which is an outcome from T5.3. The AI Toolkit v1 is publicly available online at https://alameda.catalink.eu/ and currently hosts six (6) AI-powered modules from WPs 3 and 4 for neurological and brain disorders relevant to PMSS. The second and final version of the toolkit, which will be based on feedback from the project pilots, will be presented in D5.5 (due M30).

Additionally, this report also presents the latest work within this task on defining and planning the necessary steps towards implementing the ALAMEDA AI Toolkit and the overall ALAMEDA platform integrating the individual ALAMEDA components.

With regards to the AI Toolkit modules, the next steps include the following features to be added in their final version:

- **Conversation Sentiment Analysis Toolkit** (WP4, partner: UNIC): We are planning to extend the model to also support the Romanian and Italian languages. This extension will increase the vocabulary size from 183k tokens to 310k tokens (indicative number from current experiments). Furthermore, we are investigating the potential of extending the current model to understand the sentiment polarity at a conversation level, which is currently at an experimental stage.
- **Facial Emotion Recognition Toolkit** (WP4, partner: NTNU): The vision transformer-based (ViT) model will be extended to predict six/seven facial expressions, while conducting training and testing over a combination of different publicly available datasets.
- **Gait Analysis Toolkit** (WP4, partner: NTNU): The next version of GA will include the characterization activities performed by the user (patient) taking into consideration more specific gait parameters, such as cadence, stride lengths/time, step lengths/time, etc. Such spatiotemporal parameters can be used to assess/analyse the speed, balance, symmetry, or stride-to-stride variability in PMSS patients. Furthermore, the next version of GA will be based on combined sensor data from waist-worn accelerometers (smart belts), wrist-worn accelerometers (smart bracelets), and pressure sensors (smart insoles) to predict the severity of PD, relapse risk in MS, and functional independence after Stroke.
- **Semantic Knowledge Graph** (WP4, partners: CTL & CERTH): The next version of the SemKG will include richer schema information, relevant to all the ALAMEDA-related disorders, as well as a wealth of anonymized aggregate data samples.
- **Sleep Monitoring and Assessment** (WP4, partner: ENO): Sensors have been installed and data are being collected. The classification of PD patients will require work on new algorithms for patients' classification. The assessment of PD based on their sleep quality parameters is in progress with the PD clinicians (Univ. of Athens).

It should be noted that according to the ALAMEDA DoA, the second and final version of the toolkit, which will be delivered with D5.5 on M30, will provide the AI-enabled modules along with the accompanying documentation and resources which will ultimately be available through the ALAMEDA Innovation Hub (WP7).