Ref. Ares(2021)7989376 - 27/12/2021

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

Deliverable D5.1

Title: Description Methodology & Design Principles of ALAMEDA AI Toolkit Version 1.0

Dissemination Level:	**PU**
Nature of the Deliverable:	**R**
Date:	**22/12/2021**
Distribution:	**WP5**
Editors:	**ICCS**
Reviewers:	**CTL, CERTH**
Contributors:	**ALL**

Abstract: This deliverable is the outcome of the work performed within task T5.1 - Development of Methodology for Building the Open ALAMEDA AI Toolkit. The design of the ALAMEDA infrastructure architecture and all processes for the support of the dedicated AI Toolkit is presented in detail, along with all relevant concepts, components, and interactions. Starting from the partners' vision and the ideal goals the chosen methodology allows all combined knowledge and expertise to be blended into the initial design and support the iterative development of all necessary parts throughout the project duration. Medical protocols and requirements are analyzed and transformed to critical technical requirements, which in turn lead to technical specifications of the first full-scale pilot prototypes. All views of the architecture are presented, always under the umbrella of patient data protection, GDPR, and security guidelines carefully drawn right from the beginning of the project in D1.2. Appropriate modifications of robust methodologies enable us to include the patients in the process, thus maximizing the adoption probability of the research methods and enhancing the people's quality of life.

Disclaimer

This document contains material, which is copyright of certain ALAMEDA consortium parties and may not be reproduced or copied without permission. The information contained in this document is the proprietary confidential information of certain ALAMEDA consortium parties and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information in this document may require a license from the proprietor of that information.

Neither the ALAMEDA consortium as a whole, nor any certain party of the ALAMEDA consortium warrants that the information contained in this document is capable of use, or that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using the information.

The contents of this document are the sole responsibility of the ALAMEDA consortium and can in no way be taken to reflect the views of the European Commission.

Revision History

Date	Rev.	Description	Partner
07/07/2021	0.1	First Draft	ICCS
09/10/2021	0.2	Second Draft	ICCS, WCS
23/10/2021	0.3	NEW First Draft	ICCS, CTL
1/11/2021	0.4	Updated TOC & Structure Details, Datasets, Asset Inventory, Conversational Sentiment Analysis	ICCS, WCS, CTL, UNIC
5/11/2021	0.5	Minor updates	ALL
15/11/2021	0.6	Minor updates	ALL
18/11/2021	0.7	Security, Open-source tools, references	PLU
30/11/2021	0.8	Architecture – Reqs, Views	WCS, CTL
16/12/2021	0.9	Prefinal, ready for internal review	ICCS
23/12/2021	1.0	D5.1 v1.0	ALL

List of Authors

Partner	Author
ICCS	Konstantina-Maria Giannakopoulou, Ioanna Roussaki, Panos Tsakanikas, Stavros Xynogalas
WCS	Dimitrios Karamitros, Hara Stefanou, Elena Avatangelou, Minas Badounas, Ioannis Ladakis
UPB	Alexandru Sorici
NTNU	Muhammad Sajjad
UNISYS	Ilias Aliferis, Athanasios Fameliaris
CTL	Efstratios Kontopoulos, Konstantinos Avgerinakis
ENO	Iphigenia Kapsomenaki
UNIC	Ioannis Katakis, Chloe Chira
PLU	Igino Corona
CERTH	John Tikas, Christoniki Manga-Nteve, Stelios Zygouris

Table of Contents

Index of Figures

Index of Tables

Glossary

Abbreviation	Full name
A2DP	Advanced Audio Distribution Profile
AI	Artificial Intelligence
AIH	ALAMEDA Innovation Hub
ANNs	Artificial Neural Networks
API	Application Programming Interface
AU	Action Unit
BLE	Bluetooth Low Energy
BU-3DFE	Binghamton University-3D Facial Expression
CI/CD	Continuous Integration/Continuous Delivery
CIS-PD	Clinician Input Study in Parkinson’s Disease
CMU-MOSEI	Carnegie Mellon University-Multimodal Opinion Sentiment and Emotion Intensity
CMU-MOSI	Carnegie Mellon University- Multimodal Corpus of Sentiment Intensity
CNN/ConvNet	Convolutional Neural Network
CSAT	Conversational Sentiment Analysis Toolkit
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
DL	Deep Learning
DL-FER	Deep Learning based Facial Expression Recognition
DL-GA	Deep Learning based Gait Analysis
DNN	Deep Neural Network
DREAM	Dialogue for Reverse Engineering Assessments and Methods

EDSS	Expanded Disability Status Scale
EEG	Electroencephalography
EMA	Ecological Momentary Assessment
FACS	Facial Action Coding System
FAQ	Frequently Asked Questions
FEA	Facial Expression Analysis
FER	Facial Expression Recognition
FERA	Facial Expression Recognition and Analysis
FoG	Freezing of Gait
FUNC/NON-FUNC Reqs	Functional/Non-Functional Requirements
GA	Gait Analysis
GDPR	General Data Protection Regulation
GEMEP	GEneva Multimodal Emotion Portrayals
GPU	Graphics Processing Unit
GUI	Graphical User Interface
H&Y	Hoehn and Yahr
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMU	Inertial Measurement Unit
IoT	Internet of Things
IT	Information Technology
ITI	Informatics and Telematics Institute
JSON	JavaScript Object Notation

JSON-LD	JavaScript Object Notation for Linked Data
KG	Knowledge Graph
kNN	k-nearest neighbours
LCGs	Local Community Groups
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
LTT	Line Tracking Test
MDS-UPDRS	Movement Disorder Society-Sponsored Revision of the Unified Parkinson’s Disease Rating Scale
MEAA	Mood Estimation Android Application
MFIS	Modified Fatigue Impact Scale
MJFF	Michael J. Fox Foundation
ML	Machine Learning
MoSCoW	Must-have, Should-have, Could-have, Won’t-have
MS	Multiple Sclerosis
NIR	Near Infrared
NN	Neural Network
NREM	Non-Rapid Eye Movement
OCR	Optical Character Recognition
OS	Operating System
OWL	Web Ontology Language
PCA	Principal Component Analysis
PCO	Patient-Centered Outcome
PD	Parkinson’s Disease
PDQ-8	Parkinson’s Disease Questionnaire-8

PHP	Hypertext Preprocessor
PHQ-9	Patient Health Questionnaire-9
PMSS	Parkinson’s, Multiple Sclerosis and Stroke
PRO	Patient-Reported Outcome
PVTC	Predictor Variable Time Series Classification (module)
PWD	Password
PwMS	People with Multiple Sclerosis
RBF	Radial Basis Function
RDF	Resource Description Framework
REM	Rapid Eye Movement
REST/RESTful	Representational State Transfer
RNN	Recurrent Neural Network
SDK	Software Development Kit
SDLC	Software Development Lifecycle
SemKG	Semantic Knowledge Graph
SOAP	Simple Object Access Protocol
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
SSID	Service Set Identifier
SSL	Secure Sockets Layer
SVM	Support-Vector Machine
TCL	Tool Command Language
TLS	Transport Layer Security
TUaG	Timed Up and Go

UI	User Interface
UPDRS	Unified Parkinson’s Disease Rating Scale
UPDRSm	Motor Section of the Unified Parkinson’s Disease Rating Scale
UTC	Coordinated Universal Time
VarCl	Predictor Variable Classification
VIS	Visible Light
VK	Virtual Keyboard
VST	Virtual Supermarket Test
W3C	World Wide Web Consortium
WP	Work Package
XGBoost	Extreme Gradient Boosting
XP	Extreme Programming
YAML	Yet Another Markup Language / YAML Ain’t Markup Language

Executive Summary

ALAMEDA is an ambitious interdisciplinary project, aiming to transform the entire future of medical research supported by advanced Internet of Things (IoT) technology and data science, especially Big Data and AI. Prediction of the temporal evolution of a patient's disease status as early as possible is one of the most critical desires of medicine worldwide. Until now, a doctor would need to have access to a particularly large number of patients and cooperate with a team of numerous experienced researchers with expertise on mathematics, statistics, information technology (IT), artificial intelligence (AI), and even electronics, in order to stand a chance to perform efficient research on real-time monitoring and predicting critical variables of a specific disease. The large-scale adoption of IoT technology has enabled the efficient and unobtrusive collection of a multitude of data types from large cohorts of patients. At the same time, global open data strategies have allowed access to datasets that can support meaningful research results, while data science and the advanced tools created for Big Data enable efficient processing of all available information. ALAMEDA is working on the creation of a state-of-the-art AI-powered toolkit that will offer significantly enhanced support services to small research groups with the fastest possible learning curve. This should change the order of magnitude of the people able to make a difference in medical research, as well as the time needed to extract significant results.

This is the first version of the report from the work performed within Task 5.1, entitled "Description Methodology & Design Principles of ALAMEDA AI Toolkit Version 1.0". To enable the creation and adoption of such an advanced toolkit we initially carefully design the infrastructure it will be based on, satisfying the requirements from a long chain of people involved, starting from the patients, their families and caregivers, and including all kinds of medical and other professionals. The participants of this project, though, are not standard IT-based companies with IT architects, developers, and testers. Standard processes and methods need to be appropriately modified to address the complicated principles of cooperation among such a diverse group. Right from the start it was obvious we would need to follow an iterative process with delays for the propagation of information along the full chain of interested parties, which accounts for the particular initial chosen work schedule and the relatively large number of reserves and contingency plans. The goal is to put all gathered experience and expertise in good use. The results of project MULTI-ACT[1] provided a good starting point and an efficient way to start moving forward, but it took the combined effort of the entire extended group of researchers and patients to figure out the next steps for maximum effect. As expected, the work on D5.1 revealed dependencies and prerequisites of various tasks, which could not have been fully specified beforehand. All partners were involved in Task 5.1, beyond strict effort and official participation according to the Grant Agreement, in order to set a strong foundation for all technical work, using this deliverable as ALAMEDA's "snow piercer". Being on a tight schedule and in order to make sure the pilots will be ready to start on time, we considered as the optimum choice to use multiple linked inter-work package (WP) documents simultaneously, depending on WP Leaders and additional key people to filter and coordinate

[1] https://www.multiact.eu/

all information flows. The greatest challenge was the inclusion of Local Community Groups (LCGs) in the loop, which will be the greatest success once fine-tuned.

This document is the technical root of our specifications and designs, in a tree of multiple connected relevant documents describing from medical protocols to business models. All relevant documents will be used as live documents throughout the entire first iteration, constantly updated, gathering all critical input, to be consolidated and integrated finally into the official blueprints for the ALAMEDA Innovation Hub (AIH) and its advanced services. The making of D5.1 was the first actual attempt of deploying our chosen methodology, as we had to clarify medical and technical requirements in greater detail, to the point where we could use them to elaborate on our project's concepts and design an architecture of the infrastructure and the toolkit itself. Actors and use cases referring to medical protocols and technical components were gathered from all relevant WPs, while the work methodology was customized to optimize cooperation and quality of results. Components and processes were streamlined and defined in relevant linked documents in the dedicated WPs, avoiding conflicts and overlapping. What we have achieved is a full-fledged live technical blueprint tree, which will be our map and handbook throughout ALAMEDA. All development tasks' initial specifications will be based on this document, while all information will be back-propagated from each branch to the root as needed and will be sent right back to the leaves of the tree enriched and enhanced. The views presented in this document will also be updated in real time, while all cooperation will follow the guidelines presented in the data protection plan (D1.2). The methodology and design principles revolve around the pilots and the patients, the first priority of this project.

1. Introduction

This deliverable is the outcome of the work performed within task T5.1 - Development of Methodology for Building the Open ALAMEDA AI Toolkit. This task's goal was to define the concepts behind the toolkit in terms of functional characteristics, design approach and expected outputs. PMSS (Parkinson's, Multiple Sclerosis and Stroke) AI services were validated with respect to their principles, goals, and specs for the provisioning of the ALAMEDA AI Toolkit. ALAMEDA AI methods were defined and prioritized, as were supported integration methods, e.g. Representational State Transfer (REST)/ Simple Object Access Protocol (SOAP)/Kafka streams processing, and constraints. Finally, T5.1 described appropriate solutions and designed the Toolkit so that it supports all our models and methods from WP3 and WP4. Figure 1 offers a diagrammatic representation of the interplay between T5.1 and the rest of the relevant tasks and WPs.

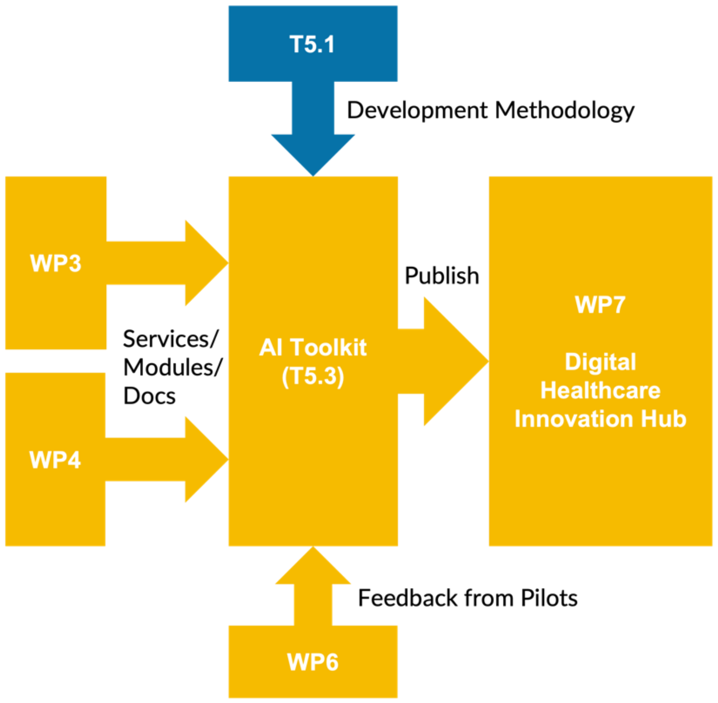


Figure 1. Interrelationship between T5.1 and other WPs and tasks within ALAMEDA

ALAMEDA intends to develop and offer the following:

- anonymized datasets from the trials,
- software modules and apps for the pilots,
- big data analytics tools,
- general and specialized neural network (NN) models (a set of basic simple neural networks with various learning methods and instructions, without the need to offer any data to a cloud,

especially critical for EU scientists vs the US) that may prove to be particularly efficient for specific use cases, like prognosis or anomaly detection etc.,
- network or catalogue of stakeholders in the domain,
- potentially a guide for new pilot instantiation, perhaps even as “extensions" of the project for future medical use-cases.

Two development cycles are planned (due M18 and M30, respectively), the latter based on feedback from the project pilots (WP6). As it is still too early to know the exact resources, timeframes, and all relevant factors, we need to choose an appropriate development methodology, based on what we do know. We have enough resources to build advanced tools, accompanied by instructions and examples for an acceptable learning curve. The more issues we manage to clarify beforehand, the better, as our course will be steadier, the deadlines will be more reasonable, and the results will be more reliable. After reaching an initial common understanding about the main project issues, we will be able to choose the most appropriate methodologies, and adapt them to our needs and priorities.

The structure of this document follows the flow of work and offers the root for the tree of blueprints and relevant live documents for every development path of the project. The participants’ vision is immediately presented in Section 2, in order to present the “big picture” and offer perspective. Then, in Section 3, appropriate modifications are applied to state-of-the-art methodologies, in order to reach an optimal customized methodology for ALAMEDA. The requirements are recorded in Section 4, covering all aspects of the work and all interested parties, but at the same time revealing the technical features needed to support the pilots. Technical specifications and architectural views are described in Section 5, documenting the design of both the pilot infrastructure and the AI toolkit. As shown in Section 6, special consideration has been given to data protection and security issues in operations and project-wide cooperation, due to the sensitive nature of patient data involved. Finally, Section 7 draws conclusions and links to the next steps.

This version of the document will be submitted, according to schedule, on M12. After that a live version of the document will be used to gather updates and important issues, until M20, when the second version of the deliverable will be finalized and submitted.

2. Vision

Our vision for the ALAMEDA AI Toolkit is to constitute a user-friendly "gateway" for interested third-party stakeholders and the overall community wishing to use the ALAMEDA AI-powered services developed within WP3 and WP4. The toolkit will ultimately be available through the AIH (WP7) and will take the form of a web-based platform for hosting the AI-enabled services and modules developed within WPs 3-5, along with the accompanying documentation.

The ALAMEDA AI Toolkit will initially be used by the project's pilot applications, such as the Experts' Dashboards aiming to facilitate doctors to draw intelligent conclusions from available data, in order to explore potential relations and dependencies among variables of interest. Building on the expertise of the ALAMEDA partners and the experience gained from the project, though, our overarching ambition is to supply a wider range of medical doctors, researchers, and healthcare professionals of various levels of competence and technical knowledge, by providing them with an arsenal of powerful AI-based tools for monitoring and predicting the progress of brain diseases.

The design of the AI toolkit is a multi-level process that starts with data collection and ends with the integration of different trained Machine Learning (ML) and Deep Learning (DL) models. Proper data collection, pre-processing, annotation, benchmarking, training of ML/DL models over collected annotated data are very important building blocks towards designing the modules of the AI toolkit (see Figure 2).

Level 1 - Data Collection: Properly collected data improves the effectiveness and prognostics accuracy of any ML-based AI application. Therefore, data collection plays the role of the "spinal cord" in the development of such applications. Data acquisition, which is required by ML/DL algorithms must accurately mimic the medical protocol. And, as suggested by the title itself of the ALAMEDA project ("*Bridging the Early Diagnosis and Treatment Gap of Brain Diseases via Smart, Connected, Proactive and Evidence-based Technological Interventions*"), the collected data must carry all the required patterns and sequences necessary for prognosis and diagnosis. Hence, the following principles must be adopted towards aiming the project's objective: suitable number of samples per class/predictive variable, data collection of multiple time series experiments with varying degree of frequency, annotation and proper benchmarking of the collected data. These activities are part of T4.2.

Level 2 - Data Annotation and Pre-Processing: The prediction accuracy of ML/DL models completely depends on the data annotation: the more accurate the data annotation, the more precise the prognostic capabilities of the trained ML/DL models. In addition, via the proper annotation, appropriate pre-processing of the annotated data augments the predictive quality of AI algorithms. Hence, it is very important to make sure that the annotation was carried out keeping in view the targets of prognosis and diagnosis set by the medical experts.

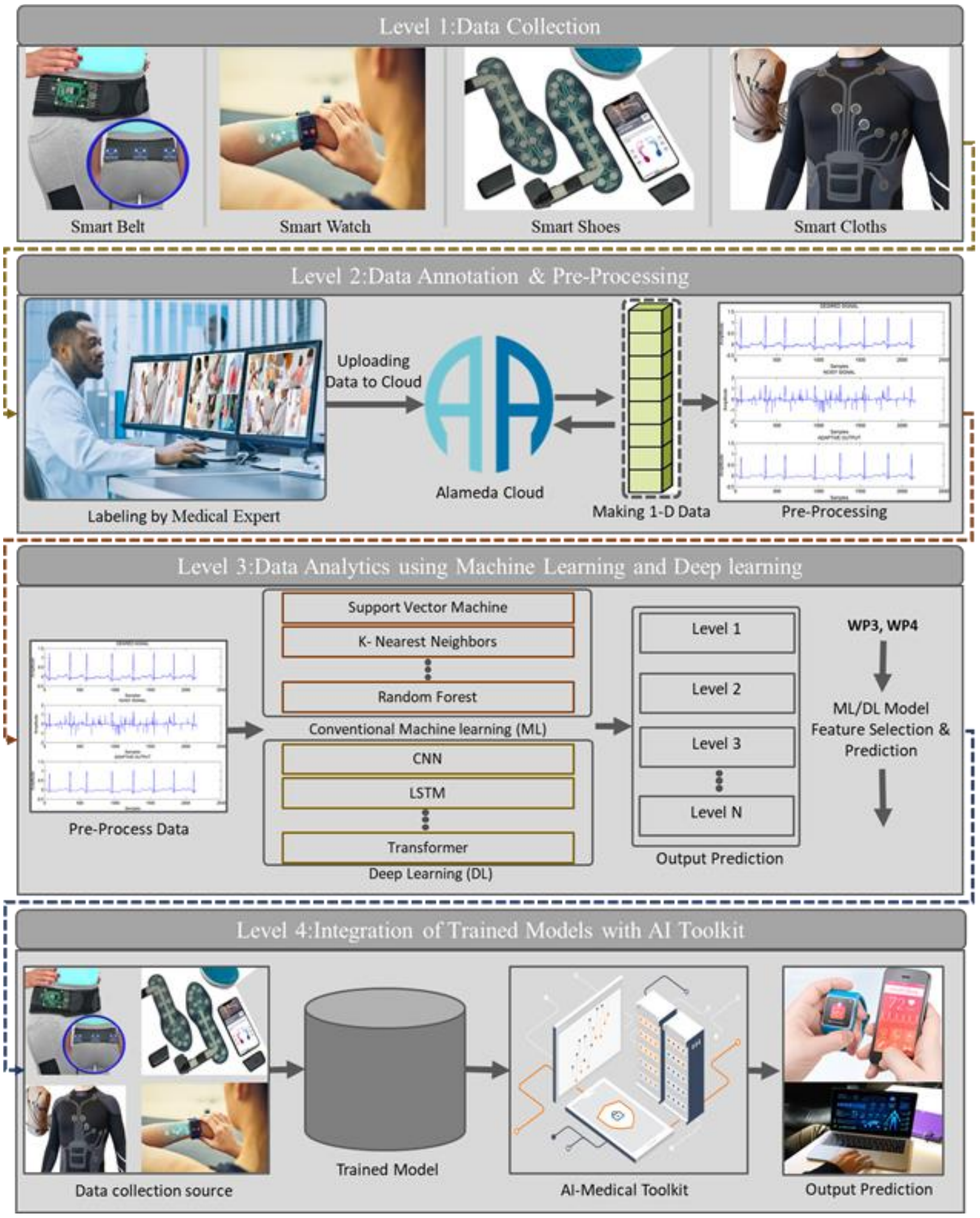


Figure 2. Core building blocks of the AI Toolkit

Level 3 - Data Analytics using ML/DL: At this stage, different ML and DL models are trained over the data annotated at the previous stage. The accurate prognosis and diagnosis can be accomplished via properly structured ML/DL models fed with the well annotated data. ML/DL models can be trained over the various modalities (e.g., wearable sensors, images/videos, and speech) to diagnose and prognose the patient conditions according to the prediction defined by the medical experts. Various ML/DL concepts, such as convolutional neural networks (CNNs), CNN-based transfer learning, generative adversarial networks (GANs), reinforcement learning, and sequence learning, can be incorporated to achieve the goals of bridging early diagnosis and treatment. Activities in this and in the previous level are parts of T4.4.

Level 4 - Integration of Trained Models with AI Toolkit: All the modules (per ALAMEDA component) mentioned thus far should be integrated into the AI toolkit to work collectively and effectively, wherever necessary. The role of this level is to provide mutual compatibility and cohesiveness among the different modules, to satisfy the project objectives. Activities in this level are part of T5.3.

3. Methodology

Past successes and failures lead to guidelines and methods that help us make the best of current situations. This section presents dominant methodologies and their features, explaining how we customize them within ALAMEDA, in order to extract the best results, based on the project's own specific needs and particularities.

The purpose of a Software Development Lifecycle (SDLC) model is to support software development in a systematic, time-effective and cost-effective way, enabling at the same time quality maximization. It refers to a continuous process which takes place from the moment that a project launches till it is completed and the produced software components are no longer exploited. In our case this timeframe extends to cover not only the project duration, but also the exploitation of the AIH and the surrounding ecosystem we will release after the end of the project.

3.1 Popular SDLC Models

There are various SDLC methodologies that cover multiple development needs and demonstrate different advantages and disadvantages. Despite their deviations and independently of the project they are applied in, all SDLC models consist of some common phases:

1. **Planning and requirements analysis:** This is the first phase where the problem and the objectives of the project are defined. A schedule is proposed for the project and its corresponding upcoming activities. The requirements for the final product are also defined after discussions with the involved stakeholders. This phase ensures technical feasibility and participants' understanding of the respective tasks. In this way, potential risks are identified early and problems that otherwise would affect development are overcome.

2. **Design:** During this phase, developers based on the requirements collected in the previous stage, design the architecture of the project. More specifically, they outline the technical details involved in the whole SDLC and the respective technical questions that arise are discussed by the stakeholders. The initial system requirements are transformed into a more logical structure, that supports the definition of all the technologies that will be used, the interfaces that will be build, the databases, as well as the operation and maintenance plans, among others.

3. **Coding and development:** This phase is the core of the SDLC, as programmers, based on the design document and the requirements previously set, select the most suitable programming languages and actually code the planned solution. The programming stage includes algorithms development, source code writing, compilation and debugging.

4. **Testing:** After developing the solution, testers need to test the produced software to make sure that there are no bugs or defects in the code, while at the same time it meets the requirements and quality standards, previously set. This stage includes both functional (e.g., unit testing, integration

testing, acceptance testing etc.) and non-functional testing (load testing, scalability testing, usability testing etc.).

5. **Deployment and maintenance:** This is the final phase of SDLC when the produced software solution has no more critical issues and is ready for release. Usually, the final product is initially loaded to testing or staging environments to enable the detection of potential final mistakes and in the end, is deployed to the production environment for exploitation from the end-users. After its release to the market, the product enters a live software maintenance environment, where the development and operations (DevOps) engineers and the technical support team ought to support end-users during exploitation and handle potential issues reported by them, by fixing residual bugs and upgrading several software components to keep the system up-to-date.

Below, we list some of the most popular SDLC models. We present their basic characteristics and their pros and cons. The comparative analysis that follows is based on published work in the literature from Ruparelia (2010) [1], Mishra and Dubey (2013) [2], Bhuvaneswari and Prabaharan (2013) [3], Kneuper (2017) [4] and Salve et al. (2018) [5].

3.1.1 Waterfall Model

One of the earliest and simplest SDLC methodologies is the waterfall model, which was proposed back in 1970 by Royce [6]. It is a traditional sequential and straightforward SDLC model, in which every phase has to be strictly completed before moving on to the next phase. Its name is inspired by the fact that it simulates the one-direction flow of a waterfall. More specifically, each stage has a separate project plan focusing on distinct sub-goals. It takes as input the output of the previous phase and feeds with information the next phase, in a cascading way. Conventionally, there is no process to go back to a previous phase, for making adjustments or modifications. A full restart is required in the case that a mistake needs to be corrected. Therefore, the waterfall model requires perfect understanding of the objectives of the project and a clear plan how to achieve them from a very early stage.

Thanks to its simplicity, it is easy to understand, to manage and to interpret. It also gives a high priority to the documentation and planning in early stages to prevent errors. In other words, it adopts the notions “define before design” and “design before develop”, which are valuable assets in the development process. Moreover, the non-overlapping phases make clear what should be delivered in the end of each one of them. So, it is suitable for small projects with clear objectives and stable requirements set in the beginning of the project.

From the other hand, due to its rigidity and inflexibility, it may prove ineffective with respect to both cost and time, as even very minor errors can cause costly delays. Similarly, it does not support exploratory development and does not enable quick changes that might be needed due to unforeseen problems. Residual errors are propagated to the next stages, which are responsible for their handling. Therefore, the waterfall model makes it difficult to integrate risk management. Moreover, perfect understanding of the project and setting solid requirements are not always feasible. Additionally, the initial waterfall model is not applicable for continuous maintenance and does not support integration of end-users’ feedback. Consequently, it is unsuitable for long and complex projects, in which the requirements may change later in their lifecycle. For all those reasons, the waterfall model may be considered outdated compared with other more agile methodologies. Further information about its induced issues are provided in [7]. Nevertheless, to overcome these drawbacks, Royce has also proposed some slightly modified models that enable returning to previous phases through feedback loops for amendments [6].

3.1.2 Iterative and Incremental Model

The iterative and incremental model [8] is an enhanced version of the traditional waterfall model, as it consists of many individual sequential processes, combined in an iterative way. In this case, it is not necessary to have defined the full set of requirements before starting the implementation of software components. Only a portion of them needs to be defined, to proceed with the development phase and get an initial version of the product with limited functionalities. Then, this version is reviewed to detect defects and identify further requirements and is extended in the next iteration. New improved successive versions of the software solution are produced at each phase. In other words, at each iteration, linear sequences provide increments, adding new functionalities to the previous version, until the full system is developed. Finally, we can say that according to the iterative and incremental model, the project and its objectives are divided to smaller subgoals, following the "divide and conquer" strategy.

The main advantage of this methodology is that a first version of the product, with limited functionalities, can be released early, with relatively low cost. This facilitates making and monitoring modifications and adjustments that will be integrated in later releases. In this way, high-risk and critical functions can be implemented early and have plenty of time for improvements. Additionally, errors are not propagated and stakeholders' feedback can be easily integrated in later extended versions. For all those reasons, it is wise to adopt iterative and incremental model, when requirements are expected to evolve or the development of the project all at once involves high risk or if there is a hurry to deploy an initial release to the market. We should also note that this SDLC methodology gives higher priority to the designing than the documentation.

However, the lack of knowledge of the full set of the requirements in advance may lead to increased costs. Similarly, the requirement for a fully functional baseline system in an early stage, which will be improved with increments in the future, may be strict or unfeasible in some cases. Finally, since many iterations may take place during the development lifecycle, there are not any clear milestones set for each iteration and that could lead to a very time-consuming process.

3.1.3 Spiral Model

One of the most flexible and risk-driven SDLC methodologies is the spiral model, introduced by Boehm in 1986 [9], [10]. Boehm attempted to combine the most attractive characteristics of the waterfall, the incremental and the evolutionary prototyping methodologies in one model. According to the spiral model, the project passes through 4 phases (planning, risk analysis, development-testing and evaluation), repeatedly in an imaginary spiral scheme. More specifically, in the first phase, the objectives are determined, starting on a small scale. In the second phase, alternative approaches and their involved risks are identified. The best approach is selected according to a resolving plan and is developed and tested in the next phase. In the final phase, the next iteration is planned, after the stakeholders evaluate the product and decide whether they will continue to commit to the project. The spiral model adopts the “start small, think big” notion, as each subsequent cycle builds on the baseline and brings gradual improvements. In each iteration, a prototype is produced, at the end of the risk analysis phase and the respective software is delivered at the end of the development phase. Deployment and maintenance take place only in the last cycle. Finally, it is worth mentioning that in this metaphorical spiral, radius corresponds to cost, while angle corresponds to progress.

The spiral methodology does not follow the constraining top-down approach of the waterfall model, reducing at the same time the number of its stages. Thanks to its thorough risk analysis through every iteration, it ensures risk minimization. So, it is suitable for large, high-risk and mission-critical projects, where requirements are not well-understood from the beginning or may change later in the project. It supports the customization of the product with additional functionalities and the integration of feedback or other modifications. Moreover, it corresponds to a more realistic approach, taking into consideration that stakeholders may change their mind regarding their commitment to the project. Another advantage of the spiral model is the early starting of software development.

However, development based on this model may be expensive and there is no reason for this costly approach in low-risk projects, with clear objectives and mainly stable requirements. In this case, maybe a sequential model would be more time- and cost-effective. Additionally, the spiral model is highly dependent on the risk analysis phase and if this process fails then the whole project is endangered. Therefore, risk analysis requires a great degree of expertise. Finally, there is no clear pre-defined time-management plan as spiral iterations may continue indefinitely. To conclude, some refinements are proposed in [11] and some other slightly modified models have also been introduced.

3.1.4 Prototype Model

Another popular SDLC methodology is the prototype model which refers to an evolutionary and iterative approach. In this case, initially only the prototype of the software solution is created and demonstrated to the clients. In this way, requirements are collected though live discussions and the obtained feedback is integrated to the next prototype version. All the modifications in requirements and functionalities take place in this prototyping cycle, which is repeated until the customers are satisfied with the demonstrated prototype, which corresponds to a feasible software solution. Only then, the technical team proceeds with the development of the agreed solution.

The establishment of communication channels between the development team and the customers has many advantages. Firstly, it ensures customers’ satisfaction, providing visibility and enabling them to get a good understanding of the final product and its functionalities in an early stage. In the same vein, it facilitates the collection of the requirements and the detection of missing functionalities. Therefore, it should be used when requirements are not well understood and communicated. Moreover, potential risks are identified early and refinements or modifications are easily adapted, accelerating designing. For all these reasons, the prototyping model is suitable for the development of web applications, human-computer interfaces and generally systems that involve user interaction.

Nevertheless, the high involvement of the clients may also raise some issues, as after seeing some prototypes, they may be reluctant to continue with the project or be inpatient to get the first release or ask continuously for more and more different functionalities and features. All these involve the risk of poor documentation due to potential continuously changing requirements, as well as the risk of quality compromise to get the product ready for release quickly. Finally, there is no predefined number of iterations that will be needed, so this process may prove time-consuming.

3.1.5 Agile Model

The agile methodology [12], [13] is a time-based approach, which incorporates features from the iterative and incremental and the evolutionary models. It is quite the opposite of structured and inflexible methodologies, like the waterfall model. According to this methodology, software is developed in iterations with small increments for new functionalities, aiming to fast and successive release cycles. More specifically, the development team works on sprints or iterations, each one of which has a specific short duration and a defined unordered set of deliverables, aiming to deliver working software components as early as possible and then improve them incrementally in the next iterations. In this way, more iterations and tests take place than in other popular SDLC models. So, it is becoming clear that agile model prioritizes development and not documentation. It also depends largely on customers, developers and testers real-time interaction throughout the development process. Finally, some subcategories of the agile methodology are scrum, extreme programming (XP), feature-driven development (FDD) and crystal. More information about the respective methodologies is provided in [14], [15].

The continuous improvement through iterations enables handling of issues, at the time that they arise and not at a later stage, as changes are easily integrated in the next release. Its flexibility to changes also supports risk minimization when adding new functionalities. In this way, software production is accelerated. Moreover, rapid continuous delivery of working software keeps customers satisfied. In conclusion, the agile methodology is suitable for complex and large-scale projects, especially for front-end applications, which involve users' interaction.

Despite all these advantages, its emphasis on communication and team effectiveness requires only motivated and trusted individuals to get on board. Additionally, if the customers do not communicate efficiently what they expect from the final product, there will be delays and problems in the development stage. Moreover, the attention that is paid to the development may result in poor documentation. Finally, the agile methodology is more complex and consequently more vulnerable to disruption that other models. Its demanding nature requires the expertise of senior developers.

3.2 ALAMEDA Software Development Methodology

The ALAMEDA Consortium includes many experienced senior developers and architects, but the multidisciplinary nature of the project does not allow a full agile approach. Chunks of software development will be produced following the agile methodology when they can be separated from the main flow and dealt with in a different thread. This will be the case of the ALAMEDA AI Toolkit, for example, after all input is gathered from the medical partners and the LCGs, thus leaving only the purely software-related decisions and the development itself to be pursued. The experience and excellence of the relevant partners will then allow efficient, timely, quality work to produce the best possible results in the fastest possible way, enabling the pilots to get started on time with a complete arsenal of solutions, fully operational and ready-to-use.

Prototyping is the chosen path for our main interdisciplinary tasks and decisions. From previous experience in projects like MULTI-ACT we had identified the significant gap between medical and technical partners, especially in such high specialization levels. There is always a slight translation needed when a non-IT user group communicates with software designers or developers, but things are even harder when state-of-the-art research is involved in both sides and even more so when there are patients involved, who are themselves the most critical user group communicating with both medical and technical partners at the same time. Considering this particular aspect of the project, we made sure the timeline was not affected, while also the deliverables were designed in a way that would allow stages of transition from less technical to more detailed documents, interweaved with the processes leading to the pilots. D2.1, for example, was used to gather initial requirements, but then a series of Experts Dashboards prototypes and other mock screens were introduced to achieve deeper understanding of the choices and needs of the entire community behind the pilots, before moving forward to architecture design and full-scale software development. This process has led us to this document and allowed us to set the communication channels necessary for the next steps.

Due to the nature of the project, the waterfall model is impossible to follow, but was included in the descriptions above as it is the starting point of significant parts of the other methodologies. The project will follow, as described in the Grant Agreement, a two-step iterative process as far as the main development is concerned. The basic infrastructure and the core of the AI Toolkit will be built in these two steps, featuring the spine of all pilot software and the foundations for the entire AIH ecosystem. There will be two sets of specifications and two software bundles, as planned. The rest of the software components and applications, however, will follow a spiral methodology with various numbers of cycles, making sure we will achieve the best possible results, while the main timeframe is respected at all times. The combination of the main two-step iteration with the multiple spirals will rely heavily on the expertise of key project personnel, namely the Coordinator, the Technical Manager, the Data Protection Officer and the WP Leaders, who are all experienced senior researchers and are the most critical joints that will take the weight of the entire ALAMEDA workforce. The people assigned to these positions have the experience and skills needed to facilitate difficult communication and cooperation tasks. They can function as translators, moderators, negotiators, and even as filters that will perform an initial routing of information for efficiency and full effect.

Special consideration has been given right from the beginning, making sure the necessary partners have even a little effort in tasks they would otherwise not participate in, but need to transfer knowledge and decisions to the right groups or WPs. Critical mass participation has been ensured in significant tasks like T5.1, no matter their size, duration and effort. This slight redundancy ensures that all the right people are there for important decisions, and also that all the right people are aware and can transfer information to all interested parties.

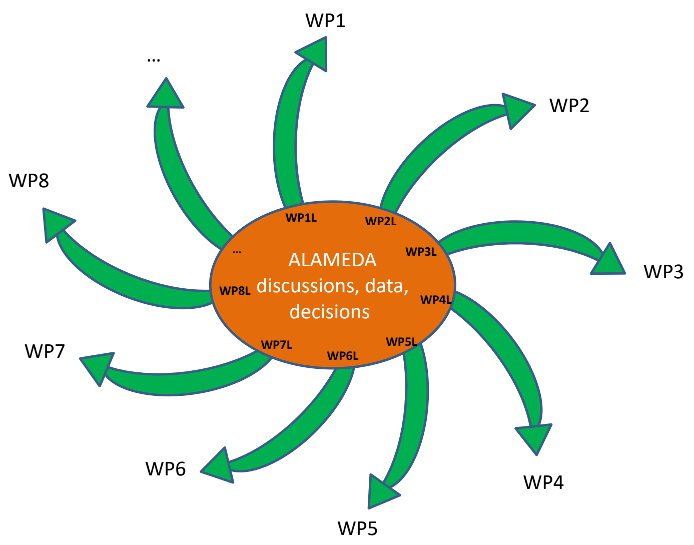


Figure 3. Critical role of WPLs within ALAMEDA

3.3 ALAMEDA Integration Methodology

Regarding integration, packaging, and documentation of the AI Toolkit modules, we will adopt the following approaches.

3.3.1 Containerization

For creating standalone "packages" of the AI Toolkit modules that won't require the installation of any additional software and/or libraries on the host machine, we will rely on the popular Docker platform. A Docker container is a virtualized runtime environment used in application development to create, run, and deploy applications that are isolated from the underlying hardware. Docker images are the starting point when using Docker, acting as a template-like set of instructions to build a Docker container. An image has everything needed to run a containerized application, including code, dependencies, configuration files, environment variables, libraries, and runtimes.

Docker images are - most commonly - created through the *Dockerfile* method. A plain text Dockerfile incorporates the specifications for creating an image using a thorough set of instructions that can be referenced in the official Docker website[2]. The user may also set up a *.dockerignore* file to exclude any files not needed for the final build. Next, the Docker *build* command is used to create a Docker image and an image *name* and *tag* are set.

[2] https://docs.docker.com/engine/reference/builder/

Once an image has been defined, it can be pushed to a Docker registry, which is a storage and distribution system for named Docker images. A Docker registry is organized into Docker repositories, where a repository holds all the versions of a specific image. The registry allows Docker users to pull images locally, as well as push new images to the registry (given adequate access permissions). For the purposes of ALAMEDA, a private registry will be deployed to host and share all pertinent images, incorporating appropriate authentication mechanisms such as pre-shared credentials.

On top of these, the capabilities of Docker Compose[3] - a tool for defining and running multi-container Docker applications - will be exploited. With Compose, the user authors a Yet Another Markup Language (YAML) file to configure application services using images (potentially from a registry). Then, with a single command, all the services from the configuration are started.

In a nutshell, the containerization process of ALAMEDA applications which we will adopt is summarized in the following three steps:

1. Definition of the application images with corresponding Dockerfiles, so they can be reproduced anywhere.
2. Registration of images to the established private Docker registry.
3. Definition of Docker Compose YAML files that assemble the services (images) of a system application, so they can be run together in an isolated environment.

3.3.2 Pre-trained ML Models

There exist several frameworks providing the tools and libraries for designing and deploying DL applications. Depending on the respective module to be added to the AI Toolkit, we will consider using one of the following:

- ML Kit[4] is a mobile Software Development Kit (SDK) from Google for Android and iOS apps, which is a cross-platform suite of ML tools for its Firebase mobile development platform and provides a set of ready-to-use APIs that could be run on-device or in the Cloud. Currently, ML Kit offers six Application Programming Interfaces (APIs) that can be deployed on the edge and are ready to use for developers with the pre-trained models, namely Image Labeling, Text Recognition (Optical Character Recognition - OCR), Landmark Identification, Face Detection, Barcode Scanning and Smart Reply. On the Cloud, Google offers 2 computer vision products that use ML to detect and classify multiple objects or text within the image and are continuously updated to achieve higher accuracy results. The first one is the Vision API, which contains pre-trained ML models, while the second one, AutoML Vision, is an easy-to-use graphical interface for the training of custom ML models.
- Core ML[5] is Apple's attempt to commodify some of the challenging ML tasks and provides four ways to integrate ML into third-party applications. More specifically, Create ML[6] is a user-friendly app that

[3] https://docs.docker.com/compose/
[4] https://developers.google.com/ml-kit
[5] https://developer.apple.com/documentation/coreml
[6] https://developer.apple.com/documentation/createml

allows building, training Apple models with custom data, and deploying them with no ML expertise required.

- For building custom-based DL models, open-source libraries such as Tensorflow and PyTorch already exist. **Tensorflow** is based on Theano and has been developed by Google. TensorFlow's high-level APIs are based on the Keras API standard for defining and training neural networks. Keras enables fast prototyping, state-of-the-art research, and production, all with user-friendly APIs. Tensor Flow models can run on mobile or IoT devices using the TensorFlow Lite convertor. TensorFlow Lite also enables on-device ML inference with low latency and small binary size. Finally, Tensorflow has a huge community behind it, which means that one can easily find various relevant resources. **PyTorch** is based on Torch and has been developed by Facebook. Unlike static graphs that are used in frameworks such as Tensorflow, PyTorch relies on dynamic computational graphs. Thus, graphs are created on the fly and debugging is extremely easy. This is particularly helpful when using variable-length inputs in recurrent neural networks (RNNs).

3.3.3 Documentation

For the AI Toolkit modules providing an API, we will adopt the OpenAPI specification v3.1.0, released in Feb'21[7]. OpenAPI is nowadays the de facto industry standard for describing, producing, consuming, and visualizing RESTful web services.

An OpenAPI specification takes the form of a YAML or JavaScript Object Notation (JSON) file and allows describing an API including:

- Available endpoints and operations on each endpoint.
- Operation parameters, input and output for each operation.
- Authentication methods.
- Contact information, license, terms of use and other information.

Besides adopting the OpenAPI specification, we will also rely on the following open-source tools built around the OpenAPI specification:

- **Swagger Editor**, a browser-based editor for writing OpenAPI specs, which we will use for turning the information received via the above templates into OpenAPI-compliant specifications.
- **Swagger User Interface (UI)** that renders OpenAPI specs as interactive API documentation, allowing the users to test the API calls directly in the browser.
- **Swagger Codegen** for generating server stubs and client libraries from the OpenAPI specs.

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

4. Requirements

For all potential users to be able to efficiently embrace the toolkit and weave it into their own work as a quality asset we need to consider the users first, their expectations, their current tools and the tasks we can help them with. The AI Toolkit will be used by the project, so the first users are the project's own medical partners, who will be using the toolkit during the pilots. After the end of the project, however, it will be integrated into the AIH, offering a valuable set of tools to the international research community, a much more diverse set of users. This section is dedicated to all users, their background, their needs, their expectations, and the added value of each tool for each user.

4.1Initial Assessment

Based on an initial general assessment, the software that will be used in the pilots should be easy to deploy at the consortium level and also at the local level by each participating institution. It should allow participating institutions to analyze their data locally without uploading them to a server or cloud infrastructure, thus ensuring compliance with GDPR and EU data privacy regulations. It needs to feature integrated visual analytics and automated reports allowing for better understanding and dissemination of research results. It should comprise an easily extendable modular design, allowing participating institutions to add their own modules and train existing modules on their datasets. After ensuring compliance with GDPR and local data privacy regulations, participating institutions need to be able to share their modules and anonymized training datasets with the whole consortium. Consortium members should be able to pool their data together to test and train models and then these models and datasets will be made available to the whole consortium.

The analysis of the prediction targets identified by ALAMEDA medical partners in relevant WPs is a good place to start. Two critical documents that indicate these prediction targets are the following:

1) **The predictor and inferred variables table** [16]

   It contains the full list of variables, both patient-generated (predictors) and resulting from clinical tests (inferred). The description of each variable allows us to determine whether it is a free numeric value or a discrete one (with values from a fixed sized enumeration). In the Inferred Vars tab the medical partners have defined prediction needs for each of the clinical test variables. The goal is to use the history of predictor variables (over the past 3, 6, 12 months) to try and predict the result of an upcoming clinical test. **All the prediction needs are in fact classification needs**, which is a critical factor for the type of AI tools (for example the type and size of neural networks and respective training techniques) we need to include. Within the predictor variables there is a subset of variables, which will themselves be the result of a classification from direct sensor input:

   - Gait pattern: **classification task** using input from smart bracelet, smart shoes, smart watch and smart belt.
   - Facial Expression Analysis (FEA): **image classification task** using input from the front facing smartphone camera.
   - Text Message Sentiment Analysis: **sentiment classification task** applied to the content of the conversation a user is having with the ALAMEDA Conversational Agent.

2) **The Intra-patient progress/regress prediction targets list** [17]

   It defines sets of cumulative conditions, which indicate a worsening or an improvement of the situation of a patient. In other words, these criteria will be used to label a subset of the patient data over a given period of time as representing a progress or regress in their situation. This is currently defined for the Stroke pilot, as it is easier to define such criteria for this affection.

Given the above, we envision the following set of tools for the project pilots:

1) Visualization

   Means to visualize multiple time series data with a varying degree of frequency per individual time series entries (one will notice in the Predictor Vars tab [18] that the rate at which inputs for each variable are taken is highly variable (from daily to weeks or months apart). One could argue that this has nothing to do with AI, but as we said from the beginning, the goal is to offer a full set of tools that would really help users produce valuable results. Visualization is critical for people searching "in the dark". A good visualization tool can offer insight and inspiration, pointing out potential anomalies or patterns and bringing us closer to a groundbreaking discovery. Many domains that have presented progress with the use of AI methods have similar needs for composite data series visualization. This is actually part of T5.2 in ALAMEDA and will probably be addressed within the Experts' Dashboard application, but it is something we should also consider in the AIH afterwards, since the code will be available.

2) Annotation

   This is another critical part of routine AI work, so relevant tools are needed to annotate subsets of inputs from one or more time series as representing a progress or regress situation. Such tools may work:

   - ❖ in a visual manner - i.e. manually select the temporal range, the variable time series and label the range as a progress or regress
   - ❖ in a rule-based manner - i.e. allow the creation of rules with temporal operators whose result is the labelling of a time series interval as a progress or regression situation

3) Focused Analysis

   - o **Gait Analysis (GA) Tools:** neural network based modules trainable on numeric multivariate time series (i.e. the sensor inputs from the smart bracelet, shoes, watch and belt in the pilots)
     - ▪ Use of the Perceiver IO [19] architecture for multi-modal input might be of help here

   - o **Facial Emotion Recognition Tools:** neural network based modules trainable on smartphone captured images of patient faces

   - o **Conversation Sentiment Analysis Tools:** neural network based modules to classify the sentiment in text from interactions with a conversational agent. Such tools are potentially useful for all medical cases that require text analysis in order to discover sentiment (e.g. psychological, mental health issues), where two modes of functionality are defined:
     - ▪ Classify sentiment in individual utterances

    - Classify sentiment in a conversation (i.e. 2 or more utterances of the user; use the context of a larger conversation history)

- **Predictor Variable Time Series Classification Tools:**
  This is the most important set of AI tools for ALAMEDA medical partners, as well as many doctors in similar areas, since it directly relates to the interest of analyzing the predictive power of wearables and patient reported questionnaires in the future performance of the patient in a medical test. The most challenging aspect in these tools is to be able to integrate both discrete and continuous value variables into the analysis. Another important aspect is extracting relevant features and selecting the most suitable ones to train a set of models. Time series feature engineering can be based on statistical measures (tsfeatures [20], Kats [21]) or machine learning algorithms (artificial neural networks-ANNs, principal component analysis-PCA etc.). The toolkit should include:
    - A classical analysis method - e.g. involving known models (e.g. Autoregressive Integrated Moving Average-ARIMA) and available libraries for time series classification (e.g. sktime [22], pyts [23])
    - Neural network based models - involving several approaches: 1D convolutional networks (convnets), attention-based models, convolutional long short-term memory (conv-lstm), RNN with gated recurrent units (GRUs) models
    - Outlier or anomaly detection features: Autoencoders, Time Series Outlier detection (TODS [24]), Unsupervised Anomaly Detection in Time Series [25], CNNs [26]

  A very-nice-to-have feature of the toolkit would be the ability to select among the inputs used for classification, i.e. from all the recorded data, train the models using only subsets of all the collected input at a time and be able to add new inputs to the model afterwards

Finally, there is a need for certain **user management** features, as well as device sharing support, especially regarding the association of data with the right patient when the devices are used by multiple patients.

Note on neural network based models:
The number of patients is great for a medical project, but mathematically it could be considered to be a small sample from which to collect data, hence the training of some neural net models may be very difficult. Therefore, we need to invest research effort into transfer learning / domain adaptation models or few-shot learning models which are able to be pre-trained on a given task (e.g. activity recognition / gait identification using a hand wrist worn bracelet) and fine-tune / adapt using few data examples from a single patient for a different task (e.g. recognize medical walking pattern using hand wrist worn bracelet and smart shoe input).

4.2 ALAMEDA Actors and Use Cases

When designing a software system, it is always helpful to write down all potential users of the system and draw all potential use cases of interest. We care who the users are, what special characteristics they may have, what they want to accomplish using the system, how they intend to use it, and of course, how we can make it easier for them. For example, a user would expect the AI Toolkit to be able to build a Neural Network (NN) from a file or save one to a file. Training an NN and using pre-trained models or storing and comparing results are other examples. However, at this stage the AI Toolkit users are only other software components and not people, so it makes sense to include such information in technical specifications or detailed software development documents later on, but not here. Doctors, for example, will only use the AI Toolkit indirectly, as will the patients and researchers during the project. We should clarify once more that the main focus of Task 5.1 and this document is the AI Toolkit for the pilots. Though we have had to clarify a lot of peripheral issues, in order to be able to analyse and design it, there is an entire dedicated WP for the critically different role of the AI Toolkit within the AIH, so we have only noted certain issues as they came up, but more details will be documented in WP7 deliverables. We have recorded interesting and useful observations and thoughts about the AIH and the transformation of the project's software that will power the entire AIH Ecosystem, which we have noted and fed into relevant discussions, but that's as far as we will go in this direction. So, to clarify, the AI Toolkit during the project is an internal software component (or rather a set of components for the project's pilots), for which there are no direct actors and use cases, while we will provide actors and use cases for it in WP7, which deals with the AIH and what we will release after the project is over.

4.3 Generalised Functional and Non-Functional Requirements

As already mentioned, projects like ALAMEDA involve people from different domains, so there is often a need for translation from one domain to another. This document starts from information coming mostly from healthcare professionals, but is to be used as input for complicated software development processes, so at this point we need to specify the requirements in a more formal manner. The requirements recorded in D2.1 present the medical point of view and the pilots' high-level requirements mostly. To proceed to the design of the AI Toolkit and the pilot infrastructure we need to start from the systems' technical requirements, so we used more technical terms, but we have left some parts in a more general form, in order to avoid creating a document that would be incomprehensible to non-IT people, as then we would not be able to use it as a base for the exchange of information among all project participants. Though this task alone is not meant to tackle every little detail of the design of all components and systems, it is the one where we need to set the foundations for all relevant work. Therefore, we made the initial analysis, discussed with all relevant and interested parties, and gathered all possibly known requirements at this stage, in order to reach the best possible understanding of what we need to build and then decide how to do it. The following tables present a classification of all provided and deduced requirements, along with some significant notes on issues like data collection and management, applications, platforms, infrastructure, and tools for the pilots. These requirements are necessary for the extraction of technical specifications and the design of the overall architecture, which will be the basis for each development task's detailed specific design.

Table 1 Overall ALAMEDA platform functional and non-functional requirements

ID	Description of Functional Requirement
FUNC-GEN-01	The ALAMEDA platform will feature the following sensors: smartphone, smart watch, bracelet, belt sensor, shoe sensor.
FUNC-GEN-02	The user will be periodically reminded by the ALAMEDA platform to charge the devices assigned to them.
FUNC-GEN-03	The data collected by the ALAMEDA platform will be of two different modalities: (1) temporally unbound (= data that can be collected from sensors or self-reported by the

	patient throughout the duration of the pilot); (2) temporally limited (= data that will be collected in a predefined window of time around the milestones defined for the trial).
FUNC-GEN-04	The data collected by the ALAMEDA platform will be of two types of frequency: (1) continuous (= the patient will provide data by wearing the sensors and consistently responding to simple lifestyle and wellbeing questions); (2) task-specific (= upon being prompted by ALAMEDA, the patient performs a specific type of activity at home).
FUNC-GEN-05	The data collected by the ALAMEDA platform will be analysed based on AI algorithms (ML & knowledge-based).
FUNC-GEN-06	The ALAMEDA platform will periodically ask the user to perform some actions to monitor their health and well-being status.
FUNC-GEN-07	The ALAMEDA platform shall provide instructions on how to perform the requested activities.
FUNC-GEN-08	The application shall display a start button to acknowledge the beginning of each exercise. Once completed, the user presses a stop button to mark the end of the exercise.
FUNC-GEN-09	The ALAMEDA platform will periodically ask the user to answer specific questions to monitor their health and well-being status.
FUNC-GEN-10	In order not to overburden the user with too many self-assessment requests, the daily life activity questions are spread out over one week.
FUNC-GEN-11	The ALAMEDA application will send requests for daily life question answering on two slots: noon and evening.
FUNC-GEN-12	Upon a request to respond to daily life questions, the user can either: choose to answer, postpone for the next slot, or forego answering any questions today altogether.
FUNC-GEN-13	To answer each question, the user will be provided with a limited set of responses.
FUNC-GEN-14	The ALAMEDA platform will suggest to the user rehabilitation programs which they should perform periodically.
FUNC-GEN-15	During the night the ALAMEDA platform will record the user's sleep patterns.
FUNC-GEN-16	Periodically, the ALAMEDA platform will aggregate the collected data and will send it to the ALAMEDA cloud to enrich the patient's statistics for the day.
FUNC-GEN-17	Physicians will be provided with aggregate data analyses by ALAMEDA, in order to adapt the suggested rehabilitation program accordingly.

Table 2 AI Toolkit functional and non-functional requirements

ID	Description of Functional Requirement
Deployment Requirements	
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.
FUNC-DEP-02	The AI Toolkit will be available as a cloud service after the project, offered via the AIH (WP7).
FUNC-DEP-03	Wherever appropriate, a module should be available for local on-premises installation.
Privacy Requirements	
FUNC-PRI-01	Compliance with GDPR must be preserved for all modules in the AI Toolkit.
FUNC-PRI-02	Data privacy regulations must be respected for all modules in the AI Toolkit.
FUNC-PRI-03	In the cases of training datasets, these need to be anonymized.
Usage & Acceptability	
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.
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 in the AIH.
FUNC-USE-03	Third parties should be able to develop new applications using the components available on the AI Toolkit in the AIH.
Documentation Requirements	
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.
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.

The following tables present the functional and non-functional requirements per AI Toolkit module. Though the module titles are intuitive, a short description of each module can be found in the next section, along with specifications, technical issues and notes.

The last column indicates the priority of each requirement, following the MoSCoW prioritization. "MoSCoW" is an acronym for must-have (M), should-have (S), could-have (C), and won't-have (W), each denoting a category of prioritization.

Table 3 DL-based Facial Expression Recognition: functional and non-functional requirements

Module: Deep Learning (DL) based Facial Expression Recognition (DL-FER)		**Layer: AI Toolkit**	
ID	**Requirement**	**Type**	**Priority**

FUNC-DL-FER-01	The DL-FER system shall receive video frames via camera from the patient. Subsequently, the face detection module shall detect the face in the video frames automatically and mark with rectangle.	FUNC	M
FUNC-DL-FER-02	Pre-processing module should pre-process the frame containing a face. Pre-processing should consist of converting the image into a pre-defined format and into the required image representation (grayscale image, binary image, etc.). In addition it should be able to modify its quality if needed. For example, converting a 24-bit per pixel image into an 8 bit-per pixel image.	FUNC	S
FUNC-DL-FER-03	The trained DL based FER model shall take the pre-processed frame and predict the patients' expression.	FUNC	M
FUNC-DL-FER-04	DL-FER system could save the predicted information on the same system for further evaluation.	FUNC	C
FUNC-DL-FER-05	Reliability is the extent to which a program can be expected to perform its intended function with precision. The DL-FER system shall be able to yield good face detection accuracy with minimal false detections.	NON-FUNC	M
FUNC-DL-FER-06	The DL-FER system shall notify the patient about potential recordings.	NON-FUNC	M
FUNC-DL-FER-07	The FER system should be usable by any user without prior training. This is achievable by adopting a 'user-friendly' design. Users also do not need to have prior knowledge of any face detection algorithm to use the system. Therefore, it is very important to make sure that the interface is user-friendly and adapted to patients with difficulties in interacting with modern technology**.**	NON-FUNC	S
FUNC-DL-FER-08	Every sub-module of the DL-FER system shall perform as it is expected. A system must operate correctly, or it provides error to its users. Correctness is the degree to which the software performs its required function. To ensure the quality of the DL-FER system, extensive testing and trial-and-error will be carried out before the system deployment.	NON-FUNC	M
FUNC-DL-FER-09	Maintainability is the ease with which a program can be corrected if an error is encountered, adapted if its environment changes or enhanced if the customer desires a change in requirements	NON-FUNC	C

Table 4 Gait based analysis: functional and non-functional requirements

Module: Deep Learning (DL) based Gait Analysis (DL-GA)		**Layer: AI Toolkit**	
ID	**Requirement**	**Type**	**Priority**
FUNC-DL-GA-01	The proposed DL-GA system shall receive signals of inertial sensors in raw form in three directions (X, Y, and Z). The inertial signal vectors (X, Y, and Z) shall reside on the same computing device.	FUNC	M
FUNC-DL-GA-02	Pre-processing module should pre-process the raw signals vectors by converting them into feature vectors.	FUNC	S
FUNC-DL-GA-03	The trained DL based GA model shall take the pre-processed frame and predict the patients' gait.	FUNC	M
FUNC-DL-GA-04	DL-GA system could save the predicted information on the same system for further evaluation.	FUNC	C
FUNC-DL-GA-05	Reliability is the extent to which a program can be expected to perform its intended function with precision. The DL- GA system shall be able to yield GA assisting the doctor in diagnosing.	NON-FUNC	M
FUNC-DL-GA-06	The DL-GA system shall notify the patient about potential recordings, keeping data privacy intact.	NON-FUNC	M
FUNC-DL-GA-07	The GA system could be usable by any user without prior training. This is achievable by adopting a 'user-friendly' design. Users also do not need to have prior knowledge of any face detection algorithm to use the system. Therefore, it is very important to make sure that the interface is user-friendly and adapted to patients with difficulties in interacting with modern technology**.**	NON-FUNC	C
FUNC-DL-GA-08	Every sub-module of the DL-GA system shall perform as it is expected. A system must operate correctly, or it provides error to its users. Correctness is the degree to which the software performs its required function. To ensure the quality of the DL-GA system, extensive testing and trial-and-error will be carried out before the system deployment.	NON-FUNC	M
FUNC-DL-GA-09	Maintainability is the ease with which a program can be corrected if an error is encountered, adapted if its environment changes or enhanced if the customer desires a change in requirements	NON-FUNC	C

Table 5 Conversation based sentiment analysis toolkit: functional and non-functional requirements

Module: Conversational Sentiment Analysis Toolkit (CSAT)		**Layer: AI Toolkit**	
ID	**Requirement**	**Type**	**Priority**
FUNC-CSAT-01	The module should be able to classify the sentiment (positive, neutral, negative) of short conversations at a sentence level and also provide quantitative scores for each class.	FUNC	M
FUNC-CSAT-02	The module will perform the necessary pre-processing (i.e. tokenization of sentences, normalization, converting text to word embeddings, and removing noise where feasible) so that the user can input the text data as it is.	NON-FUNC	M
FUNC-CSAT-03	The user of the toolkit (researchers, developers) should be able to fine tune the model on their own data.	FUNC	S
FUNC-CSAT-04	The module should not store any of the user's data or meta-data.	NON-FUNC	M
FUNC-CSAT-05	The module should support the English language.	FUNC	M
FUNC-CSAT-06	The module should support the Greek language.	FUNC	S
FUNC-CSAT-07	The module will be packaged as a downloadable docker image such that it can be used on Windows and Linux operating systems (OS). This way it can easily be integrated into other modules or be offered as a service in a server.	NON-FUNC	M

The Predictor Variable Time Series Classification (PVTC) module is the ultimate medical researcher tool, as described by our medical partners in various discussions, mostly in other WPs. The main idea is to be able to push a button and have the system automatically find potential correlations among various stored datatypes, in order to establish a connection and again automatically produce a formula for the prediction of a desired variable from existing data. Such a tool would obviously be extremely valuable, but science and technology are sadly not that advanced yet. We have documented the desires of the researchers and fully intend to design and build the best possible tools, using and wherever possible advancing state-of-the-art technology, but it needs to be clear that the recorded (in this and other documents) requirements for this module sometimes describe an ideal, unattainable set of goals. This is a clear example of not strictly technical text that we have included for internal communication reasons mostly.

Table 6 Predictor variable time series classification: functional and non-functional requirements

Module: Predictor Variable Time Series Classification (VarCl/PVTC)		**Layer: AI Toolkit**	
ID	**Requirement**	**Type**	**Priority**
FUNC-PVTC-01	The PVTC module must receive as input a **set of time series,** where each time series pertains to one **predictor variable, of one person,** identified in WP3 variables table. Each **data point** in each time series **must have** the following meta-data: • **UNIX timestamp** of data entry [mandatory] • **date in calendar format (YYYY-mm-dd)** [mandatory] • **time (HH-MM-SS) in Coordinated Universal Time (UTC)** [mandatory] • a **sessionID** of data collection [optional] • a **personID** for the person from which they were collected [mandatory] • a **sessionID** of data collection [optional] • a **disease type** (PD, MS, Stroke) [mandatory]	FUNC	M
FUNC-PVTC-02	The PVTC module **must** be able to **list** the set of sources (timeseries) it has available for a given person ID	FUNC	M
FUNC-PVTC-03	The PVTC module **must** be able to **list** the set of **target** medical tests it has available for a given person ID	FUNC	M
FUNC-PVTC-04	The PVTC module **must** be able to **select** the exact **target** medical test it wants predicted **per person ID,** as well as **the moment when it wants the classification predicted** (e.g. now, within 1 month, within 3 months, within 6 months)	FUNC	M
FUNC-PVTC-05	The user **should** be able to select **a subset** of the available input sources in a prediction task	FUNC	S
FUNC-PVTC-06	The user **should** be able to select the **time period** over which the history of the input sources is considered (e.g. the past 3 months)	FUNC	S
FUNC-PVTC-07	The prediction algorithms must provide training mechanisms that consider **a small amount** of training data	NON-FUNC	M

Table 7 Sleep quality assessment: functional and non-functional requirements

Module: Sleep Monitoring		**Layer: AI Toolkit**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SLM-01	Will assess night sleep patterns (entire sleep episode)	FUNC	S
FUNC-SLM-02	Will assess awakening in the midst of the sleep episode	FUNC	S
FUNC-SLM-03	Will assess leaving the bed (bed exit)	FUNC	S
FUNC-SLM-04	Will assess sleep apneas	FUNC	S
FUNC-SLM-05	Will assess periodic limb movements	FUNC	S
FUNC-SLM-06	Will record retiring time	FUNC	M
FUNC-SLM-07	Will record bed time	FUNC	M
FUNC-SLM-08	Will record sleep time	FUNC	M
FUNC-SLM-09	Will record wake up time	FUNC	M
FUNC-SLM-10	Will record sleep start time	FUNC	M
FUNC-SLM-11	Will record end time	FUNC	M
FUNC-SLM-12	Will record duration	FUNC	M
FUNC-SLM-13	Will record wake monitoring start time	FUNC	M
FUNC-SLM-14	Will detect sleep patterns that probably indicate health issue	FUNC	C
FUNC-SLM-15	Will correlate environmental data with sleep patterns	FUNC	S
FUNC-SLM-16	Will correlate physiometric data collected during awake state with sleep patterns.	FUNC	C
FUNC-SLM-17	The module could be able to display time series data in one graph	FUNC	C
FUNC-SLM-18	The module could be able to provide data annotation by a domain expert	FUNC	C

The Semantic Knowledge Graph (SemKG) module (introduced in D4.1) encompasses (a) the semantic model (T4.1) populated with instance data; (b) the resource description framework (RDF) triplestore persisting the populated model; (c) the Semantic KG API (T4.3/T4.4) for retrieving information from the populated model. The following tables present data related requirements.

Table 8 SemKG: functional and non-functional requirements

Module: Semantic Knowledge Graph [SemKG]		Layer: AI Toolkit	
ID	**Requirement**	**Type**	**Priority**
FUNC-SemKG-01	The Semantic KG ontology (i.e., the semantic model within the Semantic KG module) relies on existing standards and publicly available ontologies and vocabularies.	NON-FUNC	S
FUNC-SemKG-02	The Semantic KG is interoperable with other ALAMEDA modules and via its API.	NON-FUNC	M
FUNC-SemKG-03	The Semantic KG is interoperable with third-party applications via its API.	NON-FUNC	S
FUNC-SemKG-04	The Semantic KG stores sensitive personal information of the user.	NON-FUNC	W
FUNC-SemKG-05	The Semantic KG should store or have access to data generated from the following devices: SmartWatch, Smart Insoles, Smart Bracelet, Smart Mattress, SmartPhone.	FUNC	M
FUNC-SemKG-06	The Semantic KG stores information about the following entities: (a) persons (patient, clinician, caregiver, medical practitioner), (b) places/locations (indoors/outdoors), (c) events (daily living activities, physiological measurements, ambient measurements).	FUNC	M
FUNC-SemKG-07	The Semantic KG should store the following patient-related data: mobility and general motor abilities, mental and cognitive abilities, behaviour and mood assessment data, activities of daily living, motricity abilities assessment, experimental assessment, stress assessment, sleep disorders.	FUNC	M
FUNC-SemKG-08	The Semantic KG records the following information for each event: agent, start time, end time, duration, and location (where applicable).	FUNC	M
FUNC-SemKG-09	The Semantic KG records information regarding the following types of daily living activities: food and drink preparation & consumption, housekeeping (cleaning/maintenance activities), personal hygiene.	FUNC	S
FUNC-SemKG-10	The Semantic KG records information regarding the following types of sleep-related activities: night sleep, napping, awakening in the midst of sleep, bed exit, visiting the bathroom, visiting rooms other than the bathroom, sleep apneas, periodic limb movements and micro-arousals.	FUNC	S

FUNC-SemKG-11	The Semantic KG records information regarding the following types of social interaction activities: face-to-face interaction, telephone interaction.	FUNC	S
FUNC-SemKG-12	The Semantic KG records information regarding the following types of physical activities: indoors/outdoors, incidental, and dedicated physical activities.	FUNC	S
FUNC-SemKG-13	The Semantic KG records information regarding the patient's mood.	FUNC	M
FUNC-SemKG-14	The Semantic KG can respond to a set of pre-defined, parameterizable queries in reasonable times.	FUNC	M
FUNC-SemKG-15	Besides the already asserted information, the Semantic KG can respond with inferred information as well, offering deeper insights and higher-level interpretations.	FUNC	S
FUNC-SemKG-16	The Semantic KG shall inform the user appropriately in case of errors and/or wrong inputs to queries.	NON-FUNC	S
FUNC-SemKG-17	Being a central component to the ALAMEDA platform, the Semantic KG shall be reliable and responsive.	NON-FUNC	S

Table 9 Fitbit sync service: functional and non-functional requirements

Module: Fitbit Sync Service		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-FIT-01	The service should allow for the association of ALAMEDA users to their Fitbit accounts through the Digital Companion application	FUNC	M
FUNC-FIT-02	The service should perform the authorization flow required for a user to allow ALAMEDA to collect his/her data on their behalf	FUNC	M
FUNC-FIT-03	The service should allow the user to associate their Fitbit trackers to ALAMEDA	FUNC	M
FUNC-FIT-04	The service should allow the user to remove their Fitbit trackers from ALAMEDA	FUNC	M
FUNC-FIT-05	The service should sync user data in a periodic manner (e.g. every 1 hour)	FUNC	M
FUNC-FIT-06	The service should include a force sync mechanism	FUNC	M
FUNC-FIT-07	The service should allow for backwards several days sync in case a user has not synced their devices with the Fitbit API recently	FUNC	M

FUNC-FIT-08	The service should store physical and activity related data locally, in a dedicated local (to it) database	FUNC	M
FUNC-FIT-09	The service should allow for quick retrieval of activity information from its local database	FUNC	M
FUNC-FIT-10	The service should be able to store activity related data in the data management layer via the SemKG API.	FUNC	M
FUNC-FIT-11	The service should integrate with the Digital Companion application	NON-FUNC	M
FUNC-FIT-12	The service should be able to handle an increasing number of users	NON-FUNC	M
FUNC-FIT-13	The service should be able to support many concurrent users syncing their data, working in a non-blocking manner	NON-FUNC	M
FUNC-FIT-14	The service should be able to perform a full user update cycle in a small amount of time	NON-FUNC	M
FUNC-FIT-15	The service should be able to store user registration and Fitbit related authorization information using encryption	NON-FUNC	S

Table 10 Smart Bracelet Data Collection Service: functional and non-functional requirements

Module: Smart Bracelet Data Collection Service		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SB-01	The service must allow for the association between the Bracelet ID and the ALAMEDA user that wears it (using their personID).	FUNC	M
FUNC-SB-02	The service must allow for the configuration of the wear method of the bracelet: sample frequency, wear position (left wrist, right wrist, left arm, right arm, left thigh, right thigh, and waist).	FUNC	M
FUNC-SB-03	The service must be able to store raw data locally on the device for up to 2 weeks.	FUNC	M
FUNC-SB-04	The service must be able to store the raw data collected (for up to 2 weeks) from all devices in a database local to it.	FUNC	M
FUNC-SB-05	The service must be able to upload the Physical Activity Information and Sleep Information obtained from a user wear session to the ALAMEDA Data Management Layer using the KG API.	FUNC	M
FUNC-SB-06	The service must be able to interact with the Digital	FUNC	M

	Companion Application to retrieve annotations (start time, end time and name of a specific activity / test) of raw data stored in the local database.		
FUNC-SB-07	The service must enable a retrieval API for use with the DL based GA service to perform (i) gait recognition, (ii) balance problem detection, (iii) stroke physical therapy exercise classification.	FUNC	S
FUNC-SB-08	The service should provide an API to sync with the services for Smart Belt and Smart Insoles data collection to **align** raw data by meta-data, to enable improved classification of gait, balance issues and physical therapy exercise.	NON-FUNC	S

Table 11 Sleep monitoring sensors: functional and non-functional requirements

Module: Sleep Monitoring Sensors		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SLEEPSERVICE-01	The module must store measurements in relation and non-relation data structures as per data requirement.	FUNC	M
FUNC-SLEEPSERVICE-02	The module must handle client registration, authentication and session management.	FUNC	M
FUNC-SLEEPSERVICE-03	The module must be able to detect sleeping postures based on pressure sensor matrix.	FUNC	M
FUNC-SLEEPSERVICE-04	The module must offer raw data as taken from the device.	FUNC	S
FUNC-SLEEPSERVICE-05	The module must be able to accept data from environmental sensors	FUNC	S
FUNC-SLEEPSERVICE-06	The module should be able to retrieve data from CE certified Sleep Monitoring Sensors (e.g. Withings sleep analyzer).	FUNC	S
FUNC-SLEEPSERVICE-07	The module should be able to provide raw data from Withings sleep sensor	FUNC	S
FUNC-SLEEPSERVICE-08	The module could offer data upload from an end user	FUNC	C
FUNC-SLEEPSERVICE-09	The module should provide processed / result data to another module	FUNC	S
FUNC-SLEEPSERVICE-10	The module should provide API for data exchange between other modules	NON-FUNC	C
FUNC-	The module should reside in a cloud environment with	NON-	M

SLEEPSERVICE-11	99% availability	FUNC	
FUNC-SLEEPSERVICE-12	The module must be accessible over internet as REST services under Secure Sockets Layer (SSL)- Transport Layer Security (TLS)	NON-FUNC	M

Table 12 Smart Belt Sensor: functional and non-functional requirements

Module: Smart Belt Sensor		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SBS-01	Sensors shall be nearby a local Access Point to connect to the cloud via internet.	FUNC	M
FUNC-SBS-02	The Access Point should be configured with a service set identifier (SSID) and password (PWD) for sensors to be able to connect. SSID: can be provided on demand PWD: can be provided on demand	FUNC	S
FUNC-SBS-03	Data shall be downloaded from the cloud. First the user shall login to the cloud account, select the variable, and start downloading the data. Second, the user shall login to the email account where the data is sent from the cloud.	NON-FUNC	S
FUNC-SBS-04	Sensors shall be kept on charging when they are not in use.	FUNC	S
FUNC-SBS-05	Sensors must be partially or fully charged to be able to acquire and send data to the cloud.	FUNC	M

Table 13 Insole Sensor: functional and non-functional requirements

Module: Insole Sensor		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SI-01	The service must allow for the association between the SmartInsole ID and the ALAMEDA user that wears it (using their personID).	FUNC	M
FUNC-SI-02	The service must allow for the configuration of the sample frequency	FUNC	M
FUNC-SI-03	The service must be able to store raw data locally on the device for up to 1 day.	FUNC	M
FUNC-SI-04	The service must be able to store the raw data collected	FUNC	M

	(for up to 2 weeks) from all devices in a database local to it.		
FUNC-SI-05	The service must be able to interact with the Digital Companion Application to retrieve annotations (start time, end time and name of a specific activity / test) of raw data stored in the local database.	FUNC	M
FUNC-SI-06	The service must enable a retrieval API for use with the DL based GA service to perform (i) gait recognition, (ii) balance problem detection, (iii) stroke physical therapy exercise classification.	FUNC	S
FUNC-SI-07	The service should provide an API to sync with the services for Smart Belt and Smart Bracelet data collection to align raw data by meta-data (annotations), to enable improved classification of gait, balance issues and physical therapy exercise.	NON-FUNC	S
FUNC-SI-08	The service should provide explanations for recharging the devices after each day of wearing	NON-FUNC	S

The Smartphone camera data collection module, below, is responsible for capturing visual data (i.e., the smartphone user's face) and feeding it to the Mood Estimation Android Application (MEAA) for estimating the mood of the patient.

Table 14 Smartphone camera data collection: functional and non-functional requirements

Module: Smartphone camera data collection module		**Layer: Data Collection**	
ID	**Requirement**	**Type**	**Priority**
FUNC-SCDC-01	Data collection will happen in the background and will be transparent to the end user.	NON-FUNC	M
FUNC-SCDC-02	The data collected are aimed at estimating the mood of the patient through the detection of face expressions.	FUNC	M
FUNC-SCDC-03	The module captures input frames coming from the front camera of the smartphone.	FUNC	M
FUNC-SCDC-04	The module stores personal data and images.	NON-FUNC	W
FUNC-SCDC-05	The generated output data is stored in the data management layer.	NON-FUNC	M
FUNC-SCDC-06	The service should integrate with the Digital Companion application	NON-FUNC	S

The next tables present ALAMEDA applications' requirements. Some parts, especially the Experts' Dashboards application, are currently in a prototyping phase, as explained in the methodology section. Therefore, relevant software parts are not fully represented here, as we are following a continuous circle of development, presentation to users, feedback, and corrections, instead of recording requirements first. The significance of these parts and the need to establish a solid communication channel with the pilot participants led us to this procedure and the progress made benefitted many relevant discussions. The documentation of relevant software is part of Task 5.2 and may even include reverse engineering results for completeness and uniformity reasons.

Table 15 Digital companion application: functional and non-functional requirements

Module: Digital Companion (based on Wellmojo) mobile application		**Layer: Applications**	
ID	**Requirement**	**Type**	**Priority**
FUNC-WM-01	The application should only allow access to authenticated users	FUNC	M
FUNC-WM-02	The application should present the users with error messages in case a mobile, platform or internet connectivity error takes place	FUNC	M
FUNC-WM-03	The application should inform the user if there are currently no data available	FUNC	M
FUNC-WM-04	The application should allow the users to connect and authorise their Fitbit smart trackers	FUNC	M
FUNC-WM-05	The application should allow the users to manage their provided accounts	FUNC	M
FUNC-WM-06	The application should integrate with other ALAMEDA mobile applications via an options menu or directly (forming the Digital Companion)	FUNC	M
FUNC-WM-07	The application must store user data in an encrypted manner	FUNC	M
FUNC-WM-08	The application should be able to collect input from questionnaires	FUNC	M
FUNC-WM-09	The application should be able to store and retrieve data to and from the ALAMEDA platform respectively	FUNC	M
FUNC-WM-10	The application should present meaningful information to the users, based on disease specific criteria	FUNC	M
FUNC-WM-11	The application should present users with reminders on the steps they need to take regarding the monitoring/data collection process.	FUNC	S

FUNC-WM-12	The application should allow the users to tag specific lifestyle activities, not captured otherwise by the smart wearables	FUNC	C
FUNC-WM-13	The application should be able to present historical data (at least some 2-3 months old)	FUNC	S
FUNC-WM-14	The application should present the user with terms and conditions	FUNC	S
FUNC-WM-15	The application should provide a help/FAQ section for all allowed user interactions	FUNC	C
FUNC-WM-16	The application should provide an intuitive, easy to use interface	NON-FUNC	S
FUNC-WM-17	The application should be able to present data in a timely fashion	NON_FUNC	M
FUNC-WM-18	The Digital Companion should be optimised for responsiveness	NON_FUNC	S
FUNC-WM-19	The application should enforce industry standards with respect to security	NON_FUNC	M
FUNC-WM-20	The application should be able to collect information in an unobtrusive way	NON-FUNC	M

Table 16 Conversational agent application: functional and non-functional requirements

Module: Conversational Agent		Layer: Applications	
ID	**Requirement**	**Type**	**Priority**
FUNC-CA-01	The agent should be able to operate a conversation with the user only on topics relevant to the ALAMEDA pilots.	FUNC	M
FUNC-CA-02	The agent should provide a default response to input unrelated to the ALAMEDA project and its pilots.	FUNC	C
FUNC-CA-03	The agent should ask for clarifications if the input is relevant but it did not understand it.	FUNC	S
FUNC-CA-04	The agent should use the Conversational Sentiment Analysis module to detect the sentiment of the input.	FUNC	M
FUNC-CA-05	The agent's responses should be predefined.	FUNC	C
FUNC-CA-06	The agent should receive and send content to the android app.	FUNC	M
FUNC-CA-07	The agent should not store any of the user's input data.	NON-FUNC	M
FUNC-CA-08	The agent should support the English language.	FUNC	M

FUNC-CA-09	The agent should support languages other than English.	FUNC	C
FUNC-CA-10	The users should have to authenticate themselves before making queries to the agent.	NON-FUNC	S
FUNC-CA-11	If there is an internet connection, the agent should reply within less than 30s to the user's queries.	NON-FUNC	M

Table 17 Chatbot application: functional and non-functional requirements

Module: Chatbot Android Application		**Layer: Applications**	
ID	**Requirement**	**Type**	**Priority**
FUNC-ChatApp-01	The application will query the user when required (e.g. medical questionnaires) through push notifications.	FUNC	M
FUNC-ChatApp-02	The application should receive and send information to the ALAMEDA Ontology.	FUNC	M
FUNC-ChatApp-03	The application will allow other modules to ask questions to the user through its interface.	FUNC	S
FUNC-ChatApp-04	The application should receive and send information to the server that hosts the Conversational Agent module.	FUNC	M
FUNC-ChatApp-05	The application should store the user's input data and credentials in an internal local database.	NON-FUNC	M
FUNC-ChatApp-06	The application should support the English language.	FUNC	M
FUNC-ChatApp-07	The application should support languages other than English.	FUNC	C
FUNC-ChatApp-08	The application should work on Android phones with minimum SDK Android 8.0 (API level 26).	NON-FUNC	S
FUNC-ChatApp-09	The application UI should be simple enough for a non-expert user to operate.	NON-FUNC	M
FUNC-ChatApp-10	The users should have to authenticate themselves before using the application.	NON-FUNC	M
FUNC-ChatApp-11	The application will require an internet connection.	NON-FUNC	M
FUNC-ChatApp-12	The application should inform the user in case of internet connection error or connection issues with the servers/platforms.	FUNC	S

Table 18 Mood estimation: functional and non-functional requirements

Module: Mood Estimation Android Application (MEAA)		Layer: Applications	
ID	**Requirement**	**Type**	**Priority**
FUNC-MEAA-01	The module should detect the face of the patient in the video frame.	FUNC	M
FUNC-MEAA-02	The module should estimate the mood of the patient through visual information.	FUNC	M
FUNC-MEAA-03	The module should estimate the mood of the patient through user-input information.	FUNC	M
FUNC-MEAA-04	The module should detect the patient's behaviours and actions.	FUNC	C
FUNC-MEAA-05	The module should measure the sensory state to assess the mood of the patient.	FUNC	C
FUNC-MEAA-06	The module should provide results and reporting of assessment to any other interested ALAMEDA component.	NON-FUNC	M
FUNC-MEAA-07	The module should display any kind of interaction with the user in an easy, non-intrusive, and non-offensive way.	NON-FUNC	C
FUNC-MEAA-08	The module should notify the patient about potential recordings.	NON-FUNC	M

Table 19 Line tracking test: functional and non-functional requirements

Module: Line Tracking Test (LTT)		Layer: Applications	
ID	**Requirement**	**Type**	**Priority**
FUNC-LTT-01	The module should quantify user's performance in the LTT task	FUNC	M
FUNC-LTT-02	The module should store LTT data locally and allow for their extraction for analysis	FUNC	M
FUNC-LTT-03	The module should allow the user to associate their LTT account to ALAMEDA	FUNC	M
FUNC-LTT-04	The module should allow the user to remove their LTT account from ALAMEDA	FUNC	M
FUNC-LTT-05	The module should allow for easy and non-fatiguing assessment of hand dexterity	NON-FUNC	M
FUNC-LTT-06	The module should allow for detailed assessment of fine hand movements	NON-FUNC	M
FUNC-LTT-07	The module should provide an easy to understand, non-threatening, accurate summary of the user's performance after the completion of the LTT task	NON-FUNC	M
FUNC-LTT-08	The module should provide a secure mechanism for	NON-	M

	communicating usage and performance data to the CERTH server	FUNC	

Table 20 Virtual keyboard: functional and non-functional requirements

Module: Virtual Keyboard (VK)		**Layer: Applications**	
ID	**Requirement**	**Type**	**Priority**
FUNC-VK-01	The module should quantify parameters of user's typing activity in the mobile device	NON-FUNC	M
FUNC-VK-02	The module should upload keyboard timing sequence data to the CERTH data collection server	NON-FUNC	M
FUNC-VK-03	The module should allow the user to associate their VK account to ALAMEDA	FUNC	M
FUNC-VK-04	The module should allow the user to remove their VK account from ALAMEDA	FUNC	M
FUNC-VK-05	The module should not interfere with the use of the mobile device in any way	NON-FUNC	M
FUNC-VK-06	The module should ensure that no sensitive information regarding user's typing activity would be stored.	NON-FUNC	M
FUNC-VK-07	The module should provide a secure mechanism for communicating timing sequence keyboard data to the CERTH server	NON-FUNC	M

Table 21 Virtual supermarket: functional and non-functional requirements

Module: Virtual Supermarket Test (VST)		**Layer: Applications**	
ID	**Requirement**	**Type**	**Priority**
FUNC-VST-01	The module should quantify user's performance in the VST virtual space	FUNC	M
FUNC-VST-02	The module should upload usage and performance data to the CERTH data collection server	NON-FUNC	M
FUNC-VST-03	The module should allow the user to associate their VST account to ALAMEDA	FUNC	M
FUNC-VST-04	The module should allow the user to remove their VST account from ALAMEDA	FUNC	M
FUNC-VST-05	The module should allow for easy and non-fatiguing	NON-	M

	exploration of the VST digital space	FUNC	
FUNC-VST-06	The module should allow the user to recreate the experience of a familiar everyday activity	NON-FUNC	M
FUNC-VST-07	The module should provide easy to understand, non-threatening feedback to the user concerning their performance as they are engaged in the VST virtual task	NON-FUNC	M
FUNC-VST-08	The module should provide an easy to understand, non-threatening accurate summary of the user's performance after the completion of the VST virtual task	NON-FUNC	M
FUNC-VST-09	The module should provide a secure mechanism for communicating usage and performance data to the CERTH server	NON-FUNC	M

5. Architecture

The AI toolkit should make the AI models (ML, CNN) available to the pilots and provide a simple API – interface, so that medical partners may easily use it to make their experiments, test them on their datasets and extract their own results and assumptions. Knowing what will be developed is necessary, in order to choose the right way to proceed. Developers need to use the right tools, follow the right plan, and cooperate in a professional manner, to achieve high quality results. Based on the requirements and use cases presented in the previous sections, as well as the tools and modules available as open-source solutions, we have designed an appropriate architecture for our necessary pilot infrastructure and the AI Toolkit, which will be presented in this section. Furthermore, in view of the integration process (T5.4), we shall provide descriptions of the relevant input and output data for the modules of the toolkit. Different output data types shall be specified, where deemed necessary.

5.1 Technical Requirements and System Specifications

This section presents in detail the technical requirements and system specifications for the AI Toolkit, the data collection modules, and the ALAMEDA applications. The technical requirements are in correspondence with the specified functional requirements presented in the previous section.

5.1.1 AI Toolkit

The specifications for the following AI Toolkit modules are presented in the following sub-subsections: (a) Facial Emotion Recognition Toolkit, (b) Gait Analysis (GA) Toolkit, (c) Conversation Sentiment Analysis Toolkit, (d) Predictor Variable Time Series Classification, (e) Sleep Monitoring and Assessment, (f) Semantic Knowledge Graph.

5.1.1.1 *Facial Emotion Recognition Toolkit*

The FEA system will intelligently monitor the facial expression of the patients through a trained model deployed over cloud/server/laptop/smartphone etc. The proposed system will take as input the facial expression in the form of frames/video and will classify the face in the input frames/video based on the emotions contained in the facial expression training dataset into one of the specified categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Pain, and Neutral.

Table 22 Facial Emotion Recognition Toolkit technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-FER-01	FER should be initially trained with publicly available datasets containing the appropriate predictive variables.	The initial training of the FER models is performed via the suitable publicly available datasets.	FUNC-DL-FER-01 FUNC-DL-FER-02

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-FER-02	The initially trained FER models are fine-tuned with the annotated datasets collected through different pilots in presence of clinical partners.	The annotated datasets collected from the ALAMEDA pilots are utilised in the fine-tuning of the FER models.	FUNC-DL-FER-01 FUNC-DL-FER-02
TECH-FER-03	The trained model should be packaged in containers which are docker image pre-installed with DL frameworks along with all relevant libraries as a downloadable file and exportable to various operating system platforms such as Windows, and Linux.	The model should be able to run with minimum effort on Windows and Linux based operating systems.	FUNC-DL-FER-07
TECH-FER-04	FER should support English language.	The users should be able to utilise the tool in English language.	FUNC-DL-FER-07
TECH-FER-05	The trained model should take data directly from the same device or in real time in the form of images or continuous frames to make predictions accordingly.	FER is able to make a prediction based on real-time processing with input from the camera embedded/attached with the system.	FUNC-DL-FER-01 FUNC-DL-FER-03 FUNC-DL-FER-05 FUNC-DL-FER-06
TECH-FER-06	FER shall not store any of the user’s data or meta-data.	FER shall only process and save the prediction results and not the input data.	FUNC-DL-FER-04
TECH-FER-07	FER should perform the necessary pre-processing, i.e., cropping region of interest, resizing, normalization and removing noise where feasible.	FER is able to process input data in the form of videos/images and apply the required pre-processing.	FUNC-DL-FER-01 FUNC-DL-FER-02 FUNC-DL-FER-03

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-FER-08	The trained model should be able to classify the facial expression from an input image/video in one of the specified categories (Angry, Disgust, Fear, Happy, Sad, pain, Surprise, and Neutral).	The model returns the scores of each expression class per input image/video.	FUNC-DL-FER-03 FUNC-DL-FER-05 FUNC-DL-FER-07 FUNC-DL-FER-08 FUNC-DL-FER-09
TECH-FER-09	FER's output should be the probabilities for the available classes/labels where the highest probability of the corresponding class/label will be the predicted facial expression/emotion.	The models are able to accurately calculate the probabilities for the available classes/labels and yield a prediction result.	FUNC-DL-FER-01 FUNC-DL-FER-02 FUNC-DL-FER-03 FUNC-DL-FER-04 FUNC-DL-FER-05 FUNC-DL-FER-07

Table 23 Facial Emotion Recognition Toolkit system specifications

Component Name / ID	FEA application for monitoring neural disorders patients
Partner(s) name	NTNU
Short Description	Takes as input of the facial expression in the form of frames/video and classifies the mood into one of the specified categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Pain, Neutral, etc.
Core Functions	FEA and classification of the neurological disorder patient based on the emotions contained in the training dataset into one of the specified categories.

Related System Requirements	RAM => 4GB
	Processor => 2840 MHz

Inputs	Images or sequence of frames
Outputs	The proposed system will classify the facial expression of the patient/user according to the emotions containing in the trained data.
Operating Systems	Windows and Android operating systems
Programming languages & frameworks:	The component will be written in Python 3 and will be using the following frameworks/libraries: - tensorflow-lite - Pytorch - android studio - tensorflow - keras - flask

	- numpy - opencv - scikit-learn - matplotlib - seaborn - android gradle plugin of version 3.5.0
Exposed services	Put label over the image
User interface	Not ready yet
Database and databases tools	Not applicable
Hardware dependencies	Laptop, Tablet, and Smart phone contains minimum 4GB RAM will use as edge devices.
Hardware devices involved in any use cases the component applies to	● Desktop computers ● Laptops ● Mobiles

5.1.1.2 *GA Toolkit*

The proposed DL-based GA system will smartly analyse the gait pattern/behaviour of the patients through a trained model deployed over cloud/server/laptop/smartphone etc. Input will be received from the inertial measurement unit (IMU): accelerometer, gyroscope, and magnetometer. In the first phase of the development, different data analytics deep-learning models will be trained on publicly available datasets containing labels similar to the predictive variables specified in WP3. The trained models will be fine-tuned for the data acquired through the ALAMEDA pilots. The proposed models will analyse the overall behaviours of the patient specified by the medical experts.

Table 24 GA Toolkit technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-GA-01	GA should be initially trained with publicly available datasets containing the appropriate predictive variables.	The initial training of the GA models is performed is performed via the suitable publicly available datasets	FUNC-DL-GA-01 FUNC-DL-GA-02
TECH-GA-02	The initially trained FER models are fine-tuned with the annotated datasets collected through different pilots in presence of clinical partners.	The annotated datasets collected from the ALAMEDA pilots are utilised in the fine-tuning of the FER models.	FUNC-DL-GA-01 FUNC-DL-GA-02

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-GA-03	The trained model should be packaged in containers which are docker images pre-installed with DL frameworks along all relevant libraries as a downloadable file and exportable to various operating system platforms such as Windows, Linux, and Android.	The model should be able to run with minimum effort on Windows and android based operating systems.	FUNC-DL-GA-07
TECH-GA-04	GA shall support language (English).	The users should be able to utilise the tool in English language.	FUNC-DL-GA-07
TECH-GA-05	The trained model should take data directly from the same device in the CSV format to make predictions accordingly.	GA is able to make a prediction based on the input retrieved from the same device.	FUNC-DL-GA-01 FUNC-DL-GA-03 FUNC-DL-GA-05 FUNC-DL-GA-06
TECH-GA-06	GA shall not store any of the user's data or meta-data.	GA shall only process and save the prediction results and not the input data	FUNC-DL-GA-04
TECH-GA-07	GA should perform the necessary pre-processing in the input data, i.e., resizing, normalization and removing noise where feasible.	GA is able to process input data in the form of videos/images and apply the required pre-processing.	FUNC-DL-GA-01 FUNC-DL-GA-02 FUNC-DL-GA-03
TECH-GA-08	The trained model should be able to analyse gait of the patient from an input CSV file according to the prognostic variables.	The model returns the prediction results based on the input data.	FUNC-DL-GA-03 FUNC-DL-GA-05 FUNC-DL-GA-07 FUNC-DL-GA-08 FUNC-DL-GA-09
TECH-GA-09	GA's output shall be the probabilities for the available classes/labels where the highest probability of the corresponding class/label will be the predicted gait/pose estimation.	The models are able to accurately calculate the probabilities for the available classes/labels and yield a prediction gait/pose estimation result.	FUNC-DL-GA-01 FUNC-DL-GA-02 FUNC-DL-GA-03 FUNC-DL-GA-04 FUNC-DL-GA-05 FUNC-DL-GA-07

Table 25 GA Toolkit system specifications

Component Name / ID	GA mode based on DL for monitoring neural disorders patients.
Partner(s) name	NTNU
Short Description	Analyses the gait pattern/behaviour of the patients.
Core Functions	GA specified by the medical experts during the ALAMEDA pilots.
Related System Requirements	Processor > 2840 MHz RAM: > 4GB
Related NTNU Tasks	T4.4
Inputs	CSV file
Outputs	Predictions of different gaits
Operating Systems	Windows and Android operating systems
Programming languages & frameworks:	The system will develop in Python 3 and will use the following frameworks/libraries: - tensorflow-lite - Pytorch - android studio - tensorflow - keras - flask - numpy - opencv - scikit-learn - matplotlib - seaborn - android gradle plugin of version 3.5.0
Exposed services	Need properly annotated data
User interface	Not decided yet
Database and databases tools	Not applicable
Hardware dependencies	Laptop, Tablet, and Smart phone contains minimum 4GB RAM will use as edge devices.
Hardware devices involved in any use cases the component applies to	● Desktop computers ● Laptops ● Smart phones

5.1.1.3 *Conversation Sentiment Analysis Toolkit*

The Conversation Sentiment Analysis Toolkit will integrate ML-based modules for classifying sentiment in text from interactions with a conversational agent. The classification will be based on the context of the conversation. The model will output quantitative scores of each sentiment class it was trained on and the class with the highest confidence score will be considered the overall sentiment of the conversation. The classes with lower confidence scores can be used by the toolkit users to assess the competence of the model as well as to derive other conclusions depending on their needs. Furthermore, the AI Toolkit users will be able to fine-tune the pre-trained model on their own data to obtain higher accuracy on their specific task.

Table 26 Conversation Sentiment Analysis Toolkit technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-CSA-01	The trained model should be packaged as a downloadable docker image.	The model should be able to run with minimum effort on Windows and Linux based docker-enabled operating systems.	FUNC-CSAT-07
TECH-CSA-02	The tool should support more than one language.	The user should be able to choose between two different languages, English-Greek.	FUNC-CSAT-05 FUNC-CSAT-06
TECH-CSA-03	The model shall take as input the user's data to make predictions and discard them immediately.	The model shall not store any of the user's data or metadata. If the user needs to access previous results, they will have to re-enter their data.	FUNC-CSAT-03 FUNC-CSAT-04
TECH-CSA-04	The model should perform the necessary pre-processing, i.e., tokenization of sentences, normalization, converting text to word embeddings, and removing noise where feasible.	The model should be able to pre-process text input.	FUNC-CSAT-02
TECH-CSA-05	The model should be able to classify the sentiment (positive, neutral, negative) of short conversations (currently at a sentence level) and provide quantitative scores for each class.	The model should return the scores of each sentiment class per input text where the class with the highest score will be considered as the final sentiment of the input.	FUNC-CSAT-01

Table 27 Conversation Sentiment Analysis Toolkit system specifications

Component Name / ID	Conversational Sentiment Analysis Toolkit
Partner(s) name	UNIC
Short Description	Classifies the patient's sentiment during their interaction with an agent or a healthcare worker.
Core Functions	Classifies the sentiment of text input.
Related System Requirements	RAM: > 4GB
Related ALAMEDA Tasks	T4.3, T5.3

APIs	Not Ready Yet
Inputs	Text (String)
Outputs	The component will output confidence scores of each of the sentiment classes it was trained on (e.g., positive-80%, neutral-17%, negative-3%)
Operating Systems	Windows and Linux based operating systems
Programming languages & frameworks:	The component will be written in Python 3 and will be using the following frameworks/libraries: - flask - numpy - nltk - uwsgi - tensorflow - keras
Exposed services	Rest API Text output
User interface	No user interface
Database and databases tools	No database

5.1.1.4 *Predictor Variable Time Series Classification*

The Predictor Variable Time Series Classification Module allows medical practitioners to test the predictive capabilities of the collected patient data. The component operates as a classification model whose inputs are time series of data collected from the ALAMEDA pilots (from the devices used by patients, from the self-reported patient outcomes - questionnaires, from the interactions with the ALAMEDA conversational agent) and whose outputs are *result classes* of medical tests performed during milestone clinical visits.

Table 28 Predictor Variable Time Series Classification technical requirements

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-PVTC-01	The Predictor Variable Time Series Classification shall be able to process input in the form of time series where each time series pertains to one predictor variable.	The model successfully processes input time series in order to perform the required prediction.	FUNC-PVTC-01

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-PVTC-02	The Predictor Variable Time Series Classification shall be able to process the metadata (UNIX timestamp of data entry, data in calendar format, time in UTC, sessionId of data collection, personID, disease type) of the input in the form of time series	The model successfully processes the metadata of the input time series during the prediction generation.	FUNC-PVTC-01
TECH-PVTC-03	The Predictor Variable Time Series Classification component shall be able to query the endpoints of the AI Toolkit with which it interacts by demanding time series output between a start date and an end date in UNIX timestamps formats in UTC.	The Predictor Variable Time Series Classification can retrieve input in the form of time series from the AI Toolkit via respective endpoints.	FUNC-PVTC-01 FUNC-PVTC-05 FUNC-PVTC-06
TECH-PVTC-04	The Predictor Variable Time Series Classification shall receive classification input in either RDF or JSON for Linked Data (JSON-LD) format.	The Predictor Variable Time Series Classification can receive classification input in RDF or JSON-LD format.	
TECH-PVTC-05	The Predictor Variable Timeseries Classification should list the set of personIDs for which the Predictor Variable Time Series Classification is able to access records to the user.	The Predictor Variable Time Series Classification is providing the list of personIDs for which records are available.	FUNC-PVTC-02
TECH-PVTC-06	The Predictor Variable Time Series Classification toolkit should provide an endpoint for each personID for which it can provide records.	The Predictor Variable Time Series Classification is providing a list of endpoints for each personID for records retrieval.	FUNC-PVTC-02

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-PVTC-07	The Predictor Variable Time Series Classification should provide an endpoint that lists for each personID the predictor variables (time series types) which have been recorded for that personID.	The Predictor Variable Time Series Classification is providing a list of the predictor variables for each available personID.	FUNC-PVTC-03
TECH-PVTC-08	The Predictor Variable Time Series Classification should provide an endpoint that lists for each personID the inferred variables (time series types) corresponding to medical tests which have been performed for that personID.	The Predictor Variable Time Series Classification is providing a list of the inferred variables for each available personID that has medical tests.	FUNC-PVTC-03
TECH-PVTC-09	The Predictor Variable Time Series Classification toolkit should enable the selection of the predictor variable sources and the time bounds of the input sources for a prediction	The Predictor Variable Time Series Classification provides the means to select the desired predictor variable sources and the corresponding time bounds per source.	FUNC-PVTC-04 FUNC-PVTC-05
TECH-PVTC-10	The Predictor Variable Time Series Classification toolkit should enable the selection of the inferred variable targets and the time delay (1mo, 3mo, 6mo, 1y) of the prediction.	The Predictor Variable Time Series Classification provides the means to select the desired inferred variable targets and the corresponding time delay.	FUNC-PVTC-04 FUNC-PVTC-05
TECH-PVTC-11	The Predictor Variable Time Series Classification toolkit should provide prediction algorithms which can be trained even with a small amount of training data.	The Predictor Variable Time Series Classification provides the mechanism to train the prediction algorithms with small amount of training data.	FUNC-PVTC-07

Table 29 Predictor Variable Time Series Classification system specifications

Component Name / ID	Predictor Variable Time Series Classification
Partner(s) name	UPB
Short Description	The Predictor Variable Classification Module allows medical practitioners to test the predictive capabilities of the collected patient data.
Core Functions	The component works as a classification model. Its core functionality is to use trained prediction models to predict the result classes of medical tests performed during future clinical visits.
Related ALAMEDA Tasks	- T4.1 (for data representation) - T4.2 (for data collection) - T4.4 (for prediction model development) - T5.1 (for AI toolkit development)
APIs	See list of technical requirements (C_VarCls_TR_Interact_01 - 09)
Inputs	- Selected prediction model (see C_VarCls_TR_Input_04) - Selected predictor var time series (see C_VarCls_TR_Input_01)
Outputs	The probability distribution for the *result classes* of selected target (inferred) vars (see C_VarCls_TR_Interact_09)
Operating Systems	Linux (since this component is intended as a server-side application that may require Compute Unified Device Architecture -CUDA- processing)
Programming languages & frameworks:	Python frameworks such as pandas, numpy, pyts, sktime, scikit-learn, pytorch, keras
Exposed services	Specific endpoints not defined yet
User interface	The component is meant as a server side application / endpoint. Graphical User Interface (GUI) can be defined at a later stage around the API.
Database and databases tools	The component must use data from the SemKG component which acts as the data store for date from the pilots
Hardware dependencies	N/A
Hardware devices involved in any use cases the component applies to	Servers with graphics processing units-GPU processing (CUDA)

5.1.1.5 *Sleep Monitoring and Assessment*

The ENORA sleeping mattress collects posture information (~5 frames per minute) and transmits the data to the cloud platform. Additional sensors are also considered (environment, Rapid Eye Movement-REM states). The Sleep Monitoring and Assessment API accesses the information collected during the sleep periods of the monitored subjects. The focus is currently specifically on Parkinson’s Disease (PD) patients and the assessment results are provided by ML models that are using sleep laboratory data for training purposes (polysomnography). Specific PD sleep disturbances include insomnia, disordered breathing, sleep attacks, Rapid Eye Movement Sleep Behaviour Disorder (RBD), restless leg syndrome

(RLS). The assessment ML algorithms will aim at identifying the said disturbances and provide an assessment as response to input sensor data. Third party entities that are using compatible sensors will be able to use ML methods as a service. Sensitive user related data will be anonymized as required by the internal project procedures.

Table 30 Sleep Monitoring technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-ESM-01	The module should offer data exchanged for authorized clients.	Only authorized clients can have access to data.	FUNC-SLEEPSERVICE-02
TECH-ESM-02	The module should record monitor data sleep when a person is lying on a flat bed from sensors.	The system records data from the sensors (body posture, sleep cycles, environmental).	FUNC-SLEEPSERVICE-05 FUNC-SLEEPSERVICE-06
TECH-ESM-03	The module should analyse captured data from sensors.	The module should be able to detect aggregated data and pattern detections of posture, sleep cycles.	FUNC-SLEEPSERVICE-03 FUNC-SLEEPSERVICE-07
TECH-ESM-04	The module should offer raw or processed data to a client including the rest of the ALAMEDA components.	The module shall have data channels in a clean and secure way (REST APIs, Swagger documentation).	FUNC-SLEEPSERVICE-04 FUNC-SLEEPSERVICE-08 FUNC-SLEEPSERVICE-09 FUNC-SLEEPSERVICE-10 FUNC-SLEEPSERVICE-11 FUNC-SLEEPSERVICE-12

Table 31 Sleep Monitoring system specifications

Component Name / ID	ENORA Sleep Monitoring
Partner(s) name	ENORA
Short Description	This module has the following goals: • To receive data from mattress and environmental sensors. • To receive data from the Withings sleep sensor. • To provide data via API to another interface (GUI or system). • To provide annotation of time series data.
Core Functions	The core functions are: • Data consumer • Data producer • User authorization
Related System Requirements	Cloud based hosting, with TLS certificates.
Related ALAMEDA Tasks	T4.2

APIs	There are 3 types of APIs: • User Management (authorize, authenticate) • Collection of data from sensors • Retrieval of data from the cloud
Inputs	Raw data from sensors in the form of Key-Value pair, each pair is timestamped • ENORA sleeping mattress • Noise level sensor • Luminosity sensor • Temperature • Withings Sleep Sensor
Outputs	The ENORA sleep monitoring model will be provided to the AI Toolkit environment in the form of a Kafka stream. The option to implement it as a JSON service or to use other integration interfaces could also be considered. Either raw data (time series) or aggregated / processed data are provided.
Operating Systems	Ubuntu Server 20.04
Programming languages & frameworks:	.NET CORE 5, written in C#
Exposed services	Rest API json services
User interface	GUI is Hypertext Markup Language (HTML)/JavaScript based and Angular
Database and databases tools	MongoDB and Postgres databases are used
Hardware dependencies	WiFi Router
Hardware devices involved in any use cases the component applies to	• Servers • Desktop computers • Data acquisition devices • Network (WiFi, Ethernet)

5.1.1.6 *Semantic Knowledge Graph (SemKG)*

The SemKG module encompasses (a) the semantic model (T4.1) populated with instance data; (b) the RDF triplestore persisting the populated model; (c) the Semantic KG API (T4.3/T4.4) for retrieving information from the populated model.

Table 32 Semantic Knowledge Graph technical requirements

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SemKG-01	The Semantic KG shall comply with World Wide Web Consortium (W3C) standards.	The Semantic KG shall be fully compliant to W3C recommendations and relevant open standards.	FUNC-SemKG-01

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SemKG-02	The Semantic KG shall have no vendor lock-in.	The Semantic KG (i.e., the semantic model, the repository for persisting it, and its API) will be based on custom code and open-source libraries without any dependency on proprietary technologies.	FUNC-SemKG-01 FUNC-SemKG-02 FUNC-SemKG-03
TECH-SemKG-03	The data residing in the Semantic KG shall be accessible to other ALAMEDA modules.	The Semantic KG features a RESTful API.	FUNC-SemKG-02
TECH-SemKG-04	The Semantic KG shall take privacy and sensitive information into consideration.	The Semantic KG will not store personal data and other sensitive information.	FUNC-SemKG-04
TECH-SemKG-05	The Semantic KG shall efficiently represent all the project-pertinent data.	The Semantic KG effectively facilitates the needs of the pilots.	FUNC-SemKG-05 to FUNC-SemKG13
TECH-SemKG-06	The Semantic KG shall feature a good degree of usability.	Requests to the Semantic KG via the API shall be easily parameterizable.	FUNC-SemKG-14
TECH-SemKG-07	The Semantic KG shall have satisfactory performance and response times.	The API of the Semantic KG will have an average response time of less than 2 seconds.	FUNC-SemKG-14
TECH-SemKG-08	The Semantic KG shall perform semantic reasoning based on W3C standards.	Responses from the Semantic KG shall also include implicit information, wherever relevant. The respective result-sets will be generated based on rule-based Simple Protocol and RDF Query Language (SPARQL) reasoning.	FUNC-SemKG-01 FUNC-SemKG-15

Requirement ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SemKG-09	The Semantic KG shall feature efficient error handling.	Erroneous requests via the API will result in informative error messages returned to the user.	FUNC-SemKG-16
TECH-SemKG-10	The Semantic KG shall feature a good degree of serviceability.	Changes and upgrades to any of the elements will not require total outages.	FUNC-SemKG-17

Table 33 Semantic Knowledge Graph system specifications

Component Name / ID	Semantic Knowledge Graph (SemKG)
Partner(s) name	CERTH, CTL
Short Description	The Semantic KG plays a central role, establishing semantic interoperability between the ALAMEDA system components, and contains: (a) the populated semantic model that represents information made available via the questionnaires and the monitoring modules of the overall system; (b) the API for interacting with the populated semantic model; (c) the RDF triplestore for hosting the model.
Core Functions	Allows the other ALAMEDA components to retrieve information regarding: • behavioural interpretations and reported difficulties of the patient in the home environment, • tests, assessments, patient's clinical and experimental records in the lab environment, • sociodemographic data and information about persons, diseases, gender, educational levels, and languages, • entities and activities that take place in the context of the ALAMEDA use cases, • the type and properties of the sensors used in the ALAMEDA system, • the temporal dimension, namely, the time, duration, and information of the tasks/events taking place within the ALAMEDA context.
Related System Requirements	Heavily depend on size of the populated semantic model. Indicatively: • For 100M statements (= ~33M RDF resources): Java heap = 4GB, RAM = 6GB, HDD = 12GB. • For 200M statements (= ~66M RDF resources): Java heap = 8GB, RAM = 12GB, HDD = 24GB. • For 500M statements (= ~166M RDF resources): Java heap =

	18GB, RAM = 24GB, HDD = 60GB. • For 1B statements (= ~333M RDF resources): Java heap = 30GB, RAM = 38GB, HDD = 120GB.
Related ALAMEDA Tasks	T4.1, T4.3, T4.4, T5.3
APIs	For accessing the information residing in the Semantic KG, the module exposes a dedicated RESTful API.
Inputs	• Smart Watch (Fitbit) data • Smart Bracelet (GeneActive) data • Smartphone data • Smart Insole (Shoe pressure) • Smart Mattress data • Smart Belt data • Virtual Supermarket Test data • Line Tracking Test • Virtual Keyboard data
Outputs	Patient-related information: 1) Emotional status assessment; 2) Mental ability assessment; 3) Quality of life assessment; 4) Mobility domain assessment; 5) Sleep disorder assessment.
Operating Systems	Component is OS-agnostic.
Programming languages & frameworks:	• The semantic model is built on W3C Web Ontology Language (OWL)[8]. • The API is built on Java v14 and uses the Apache Jena[9] framework for building Semantic Web and Linked Data applications.
Exposed services	REST API
User interface	No UI provided.
Database and databases tools	The semantic model needs to be hosted on a suitable "triplestore", i.e., an ontology DBMS. For the purposes of ALAMEDA, we will be deploying the free version of the well-established Ontotext GraphDB[10].
Hardware dependencies	No hardware dependencies.
Hardware devices involved in any use cases the component applies to	No hardware devices involved.

[8] https://www.w3.org/OWL/
[9] https://jena.apache.org/
[10] https://www.ontotext.com/products/graphdb/

5.1.2 Sensor-based Data Collection Technical Specifications

The specifications for the following data collection modules are presented in the following sub-subsections: (a) Fitbit sync data collection, (b) Smart bracelet data collection, (c) Smart mattress, (d) Smart belt, (e) Insole Sensor data collection, (f) Smartphone camera.

5.1.2.1 *Fitbit Sync Data Collection*

Table 34 Fitbit sync service technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-FIT-01	The Fitbit sync service will expose REST API that allow for full user and Fitbit account management	Users will be able, via Wellmojo, to fully manage their trackers, their connection status, and all authorization aspects	FUNC-FIT-01 FUNC-FIT-02 FUNC-FIT-03 FUNC-FIT-04 FUNC-FIT-11 FUNC-FIT-15
TECH-FIT-02	The Fitbit sync service will provide scheduled runners that periodically sync users' data	Only time periods when a user has not connected their Fitbit account will not have data	FUNC-FIT-05 FUNC-FIT-06 FUNC-FIT-07
TECH-FIT-03	The Fitbit sync service will provide local storage for the management of users' raw Fitbit data	In case of data loss, recovery or for presentation purposes, raw data will be available from the Fitbit sync service	FUNC-FIT-08 FUNC-FIT-09
TECH-FIT-04	The Fitbit sync service will have REST clients for the SemKG API that allow the transfer of summarised user data to the ontology	All data collected from Fitbit service will be summarised and stored in the ontology	FUNC-FIT-10
TECH-FIT-05	The Fitbit sync service needs to perform data collection of a big number of users in an asynchronous manner	The service will be able to handle smoothly at least 500 concurrent users	FUNC-FIT-12 FUNC-FIT-13 FUNC-FIT-14

Table 35 Fitbit sync service system specifications

Component Name / ID	Fitbit Sync Service
Partner(s) name	WCS
Short Description	Fitbit sync service is responsible for the collection of the patients' Fitbit tracked data (physical activities, vital signs,

	sleep) in raw and summarized formats
Core Functions	- Facilitation of the authorization procedure required for a patient to authorize the service to collect their data on their behalf - User authorization management between ALAMEDA and Fitbit Web API - Periodic sync of user data from the Fitbit Web API daily - Periodic sync of user data from the Fitbit Web API weekly (to avoid loss of data when the tracker has not been synced manually for at most 7 days) - Management of daily summaries for physical activities, sleep, and heart rate.
Related System Requirements	Any platform running Docker or has Java 11
Related ALAMEDA Tasks	T4.2
APIs	-
Inputs	• Raw data from Fitbit Web API in JSON format
Outputs	- Raw and summarised physical activity and heart rate data in JSON format
Operating Systems	
Programming languages & frameworks:	Spring Boot framework, Java 11
Exposed services	JSON
User interface	REST API
Database and databases tools	PostgreSQL
Hardware dependencies	A server with at least 1gb of RAM
Hardware devices involved in any use cases the component applies to	Server

5.1.2.2 *Insole Sensor Data Collection*

The Insole Data Collection service is responsible for collecting data from smart insoles given to patients during the ALAMEDA pilot studies. The smart insoles are part of the set of *specific monitoring devices* (together with the smart belt and the smart bracelets) that are handed to pilot participants for a limited duration of up to 2 weeks at specific periods (see Deliverable D3.1 for details).

The smart insole service is tasked with configuring and collecting data from individual Novel Loadsol smart insoles. Configuration happens prior to a wear period. The service uses Bluetooth LE and a smartphone application to collect data from the insoles. Data from the phone is exported at the end of a wear period for collection in the Insole Local Storage Cloud. The service syncs with the WellMojo-based ALAMEDA Digital Companion to retrieve the annotations (meta-data for name of activity/exercise and its start time and end time) of specific activities or exercises carried out while wearing the insoles.

The service enables an API for data retrieval for use with the DL based Gait Recognition Service which can combine annotated data from the Smart Bracelet, Smart Insole and Smart Belt data collection services to perform gait classification, balance issue detection and physical therapy exercise classification.

Table 36 Insole sensor data collection service technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SICollect-01	The service must have a configuration interface which binds the insoles to a patientID and configures the wear method (e.g. sensor sample frequency)	The configuration service runs as a standalone application on a Windows OS machine	FUNC-SI-01, FUNC-SI-02
TECH-SICollect-02	The service must have an interface to retrieve data sent by a pair of insoles over Bluetooth to a smart phone. Retrieval happens over a USB connection from the phone to a computer.	The insoles data retrieval service runs as a parameterizable script in command line mode.	FUNC-SI-03 FUNC-SI-04
TECH-SICollect-03	Raw data collected from a pair of insoles is saved in a local cloud in CSV or Python Pandas readable binary files format indexed by personID and wear periods (start and end timestamp of the period in which the insoles were given to a patient).	The local cloud storage service implements a REST-based API to query stored data using a personID and a wear period.	FUNC-SI-04

TECH-SICollect-04	The service uses the SemKG API to retrieve annotation data (activity name, start time, end time) of specific activities, which was inserted in the ALAMEDA Data Management Layer by the ALAMEDA Digital Companion.	A SemKG service endpoint for annotated activities exists. A query with a personID parameter returns all existing activity annotations for that person.	FUNC-SI-05
TECH-SICollect-05	The service must implement retrieval service for access to annotated data.	The annotated data service retrieval implements a REST endpoint which can be queried with a personID, activity name, and session timestamps (start and end times)	FUNC-SI-07 FUNC-SI-08

Table 37 Insole sensor data collection service system specifications

Component Name / ID	Smart Insoles Data Collection
Partner(s) name	UPB
Short Description	The Smart Insoles Data Collection service is responsible with configuring Novel Loadsol smart insoles wear periods, retrieving data from individual insoles by Bluetooth LE transmission to smart phones, implementing the local cloud storage for raw data, annotating raw data with the wear session meta-data (activity name, start and end time) and enabling APIs for annotated smart insole data retrieval for use in other ALAMEDA services (e.g. the DL based GA Service).
Core Functions	• Configure smart insoles • Retrieve data from individual insoles • Annotate raw data with wear session meta-data and enable annotated data retrieval.
Related System Requirements	Ubuntu Linux 20.04 for smart insole cloud storage service. Windows 10 machine for smart insole configuration and data visualization service.

Related ALAMEDA Tasks	T4.2, T4.4
APIs	Custom API for the smart insole local cloud service for raw and annotated data access
Inputs	• Raw Plantar force data recorded by the Novel Loadsol insoles.
Outputs	- Annotated raw plantar surface data for use in Gait Recognition, Balance Issues Detection and Physical Therapy Exercise classification
Operating Systems	Windows 10 and Ubuntu 20.04
Programming languages & frameworks:	Loadsol Smartphone App Loadpad analysis software Loadpad player software (for data visualization)
Exposed services	Smart Bracelet Configuration Services; Smart Bracelet Annotated Data Retrieval Service
User interface	Windows configuration interface for Novel Loadsol Insoles; Loadsol Smartphone App
Database and databases tools	MongoDB for local cloud storage of data collected from smart insoles
Hardware dependencies	Storage server equipped having at least 100 GB of storage space available
Hardware devices involved in any use cases the component applies to	Novel Loadsol power supply dock used for charging the smart insoles

5.1.2.3 *Smart Mattress*

The prototype Smart Mattresses that will be used in the pilots are still under development, so the complete technical requirements and specifications will be provided (and probably included in relevant WP4 documents) as soon as it is complete, since there are still a few pending technical issues to be tackled.

5.1.2.4 *Smart Belt*

The prototype Smart Belts that will be used in the pilots are still under development, so the complete technical requirements and specifications will be provided (and probably included in relevant WP4 documents) as soon as it is complete, since there are still a few pending technical issues to be tackled.

Table 38 Smart belt data collection technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SBS-01	The sensors shall be mounted on the belt at the specified positions.	The belt should be worn at the waist level; leather belt or something similar must have 5 cm width max.	FUNC-SBS-01 FUNC-SBS-02 FUNC-SBS-05
TECH-SBS-02	Sensors shall be kept on charging when they are not in use. The USB port must be used for charging. The ON/OFF button must be ON before putting the sensor on a charger. The green light turns ON if the battery is fully or partially charged. Otherwise the device must be put on the USB charger. When the battery is full, the orange LED is turned OFF.	The sensor battery should be charged enough for the sensor to stay running and connected to the Internet.	FUNC-SBS-04
TECH-SBS-03	The developed system shall send data directly to the Arduino cloud in real time. All sensors' communications to the cloud use the industry standard SSL protocol for encryption. The communication board used in each sensor is the Arduino MKR which has on-board crypto-authentication chips and is further secured using X.509 certificate-based authentication.	Sensors should not store any of the data in their internal memory.	FUNC-SBS-02

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SBS-04	A Wi-Fi Access Point shall be installed for the sensors to connect to the Internet. Sensors operate at 2.4 GHz; therefore, the Wi-Fi Access Point must have this option. Sensors must be within the range of the Access Point.	The Access Point should be configured with SSID and PWD for sensors to be able to connect. Username and password can be provided to the data collection partners on demand.	FUNC-SBS-01 FUNC-SBS-02
TECH-SBS-05	Data shall be downloaded from the cloud; first the user shall login to the cloud account, select the variable, and start downloading the data Second, the user shall login to the email account where the data is sent and can be downloaded from there. A web browser must be used to login to the accounts. The downloaded data is in CSV format.	The user should have the username and password of the cloud account NTNU has on the Arduino cloud to start the download of the data. The user should have the username and password of the email account that NTNU has in order download the data	FUNC-SBS-03

5.1.2.5 *Smart bracelet data Collection*

The Smart bracelet data collection service is responsible for gathering data from the ActivInsights GENEActiv bracelets given to patients during the ALAMEDA pilot studies. The smart bracelet is part of the set of *specific monitoring devices* (together with the smart belt and the smart insoles) that are handed to pilot participants for a limited duration of up to 2 weeks at specific periods (see Deliverable D3.1 for details).

The smart bracelet service is responsible for configuring the data collection setup for each individual bracelet given to a person prior to a wear period. The service can retrieve the data stored on each device and upload it to a local cloud. The service syncs with the WellMojo-based ALAMEDA Digital Companion to retrieve the annotations (meta-data for name of activity/exercise and its start time and end time) of specific activities or exercises carried out while wearing the bracelet.

The service enables an API for data retrieval for use with the DL based Gait Recognition Service which can combine annotated data from the Smart Bracelet, Smart Insole and Smart Belt data collection services to perform gait classification, balance issue detection and physical therapy exercise classification.

Table 39 Smart bracelet data collection technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SBCollect-01	The service must have a configuration interface which binds the bracelet to a patientID and configures the wear method (body mount point, sensor sample frequency)	The configuration service runs as a parameterizable script in command line mode.	FUNC-SB-01, FUNC-SB-02
TECH-SBCollect-02	The service must have an interface to retrieve data from individual bracelets connected via USB to a computer.	The bracelet data retrieval service runs as a parameterizable script in command line mode.	FUNC-SB-03 FUNC-SB-04
TECH-SBCollect-03	For data from a bracelet collected during a wear session the service must use algorithms to extract general statistics about Physical Activity and Sleep.	The algorithms run automatically as part of the bracelet data retrieval service and output two separate CSV files for the Physical Activity Information and the Sleep Information	FUNC-SB-05
TECH-SBCollect-04	Raw data collected from a bracelet is saved in a local cloud in CSV or Python Pandas readable binary files format indexed by personID and wear periods (start and end timestamp of the period in which the bracelet was given to a patient).	The local cloud storage service implements a REST-based API to query stored data using a personID and a wear period.	FUNC-SB-04

TECH-SBCollect-05	The service uses the SemKG API to retrieve annotation data (activity name, start time, end time) of specific activities, which was inserted in the ALAMEDA Data Management Layer by the ALAMEDA Digital Companion.	A SemKG service endpoint for annotated activities exists. A query with a personID parameter returns all existing activity annotations for that person.	FUNC-SB-06
TECH-SBCollect-06	The service must implement retrieval service for access to annotated data.	The annotated data service retrieval implements a REST endpoint which can be queried with a personID, activity name, and session timestamps (start and end times)	FUNC-SB-07 FUNC-SB-08

Table 40 Smart bracelet data collection system specifications

Component Name / ID	Smart Bracelet Data Collection
Partner(s) name	University Politehnica of Bucharest (UPB)
Short Description	The Smart Bracelet Data Collection service is responsible with configuring bracelet wear periods, retrieving data from individual bracelet devices, implementing the local cloud storage for raw data, extracting general Physical Activity and Sleep Information, annotating raw data with the wear session meta-data (activity name, start and end time) and enabling APIs for annotated smart bracelet data retrieval for use in other ALAMEDA services (e.g. the DL based GA Service).
Core Functions	• Configure smart bracelets • Retrieve data from individual bracelets • Extract general Physical Activity and Sleep statistics from individual bracelets • Annotate raw data with wear session meta-data and enable annotated data retrieval.
Related System Requirements	Ubuntu Linux 20.04 for smart bracelet cloud storage service. Windows 10 machine for smart bracelet configuration service.

Related ALAMEDA Tasks	T4.2, T4.4
APIs	Custom API for the smart bracelet local cloud service for raw and annotated data access
Inputs	• Raw IMU data recorded by the ActivInsights GENEActiv braclet.
Outputs	- Physical Activity statistics - Sleep statistics - Annotated raw IMU data for use in Gait Recognition, Balance Issues Detection and Physical Therapy Exercise classification
Operating Systems	Windows 10 and Ubuntu 20.04
Programming languages & frameworks:	Python, R Python frameworks: PyGeneactiv, sleeppy R packages: GENEAread, GENEAClassify
Exposed services	Smart Bracelet Configuration Services; Smart Bracelet Annotated Data Retrieval Service
User interface	Windows configuration interface for GENEActiv Bracelets; configuration and data retrieval command line scripts in Python
Database and databases tools	MongoDB for local cloud storage of data collected from smart bracelets
Hardware dependencies	Storage server equipped having at least 100 GB of storage space available
Hardware devices involved in any use cases the component applies to	ActivInisghts GENEActiv power supply dock used for charging and reading data from the smart bracelets

5.1.2.6 *Smartphone Camera*

Table 41 Smartphone camera data collection technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SCDC-01	The patient should not feel that he/she is being monitored in order to keep his/her expressions unaffected.	The facial expression detection service should run as a background service.	FUNC-SCDC-01

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-SCDC-02	The analysis of the input frames will lead to the estimation of the mood of the patient.	The service will take as input video frames coming from the front camera of the phone and will output an estimation on the patient's mood.	FUNC-SCDC-02
TECH-SCDC-03	The output should be available to the rest of the Digital Companion components.	The service will produce as an output the mood estimation and will store it in the management layer.	FUNC-SCDC-05 FUNC-SCDC-06

The respective system specifications are included in those specified for the Mood Estimation Android Application (MEAA) presented in Subsection 5.1.3.4.

5.1.3 ALAMEDA Applications

The specifications for the following ALAMEDA applications are presented in the following sub-subsections: (a) Wellmojo mobile application, (b) Conversational Agent application, (c) Chatbot application, (d) Mood Estimation Android Application, (e) Line Tracking Test application, (f) Virtual Keyboard application, (g) Virtual Supermarket Test.

5.1.3.1 *Wellmojo Mobile Application*

Table 42 Wellmojo mobile application technical requirements

Requirement_ID	**Description**	**Acceptance criteria**	**Related FUNC/NON-FUNC reqs**
TECH-WM-01	Wellmojo should be able to seamlessly integrate other mobile applications	All other applications that form the Digital Companion will either directly be opened via a dedicated menu option or be able to communicate with them in the background.	FUNC-WM-06
TECH-WM-02	Wellmojo will facilitate the collection of data from Fitbit wearables and questionnaires	Wellmojo will gather user generated input from questionnaires, as well as facilitate the process of	FUNC-WM-04 FUNC-WM-08 FUNC-WM-12 FUNC-WM-20

		authorisation between the Wellmojo user and the Fitbit sync service	
TECH-WM-03	The Wellmojo application will present all collected information to the PMSS patients in a meaningful manner	Wellmojo shall present a dashboard based interface without including only what is important to the patients and excluding unnecessary information and elements.	FUNC-WM-02 FUNC-WM-03 FUNC-WM-10 FUNC-WM-11 FUNC-WM-13 FUNC-WM-15 FUNC-WM-16 FUNC-WM-17 FUNC-WM-18
TECH-WM-04	The Wellmojo application will be to establish a communication channel with the ALAMEDA platform.	Wellmojo shall share the gathered data in an appropriate and secure way (i.e., REST APIs, publish/subscribe models etc.).	FUNC-WM-09
TECH-WM-05	Wellmojo shall allow management of privacy and personal information	The MEAA module shall present a clear privacy policy to the patients, explaining in detail the data that will be gathered. Private and personal data will undergo encryption on the device. Access will only be allowed after appropriate user authentication.	FUNC-WM-01 FUNC-WM-05 FUNC-WM-07 FUNC-WM-14 FUNC-WM-19

Table 43 Wellmojo mobile application system specifications

Component Name / ID	Wellmojo mobile application
Partner(s) name	Wellics
Short Description	Wellmojo is responsible for the integration of all the ALAMEDA mobile applications, eventually forming the ALAMEDA Digital Companion

Core Functions	• A dashboard allowing for the presentation of patient related data in the nutrition, social, mood and physical activity aspects. • Individual (dedicated to the above categories) screens with more details and historical data. • Management of Fitbit wearable tracker • Presentation and management of (medical and application specific) questionnaires and their answers • Push notifications acting as reminders for taking actions or for presenting PMSS related information
Related System Requirements	Android 10 (API 29) and above
Related ALAMEDA Tasks	T5.2
APIs	-
Inputs	• User input while filling the questionnaires with on screen controls or through the virtual keyboard. • User profile data and credentials
Outputs	• JSON based responses to questionnaires
Operating Systems	Android OS
Programming languages & frameworks:	Ionic Framework
Exposed services	JSON
User interface	The application's interface
Database and databases tools	SQLite, encrypted local storage
Hardware dependencies	-
Hardware devices involved in any use cases the component applies to	Android Smartphone

5.1.3.2 *Conversational Agent application*

The Conversational Agent will engage patients and caregivers in cognitive and medical data collection by carefully constructed human-like interactions. The interactions will be grounded in aspects of relevance from a medical perspective, and they will be divided in four use cases: 1) answering medical questionnaires and questions about non-disease related factors, 2) collecting data about the well-being/mood of the user, 3) other intelligent components querying the user to collect information complementary to the sensors, and 4) free text. The agent will embed an empathetic personality by utilizing sentiment analysis on the user's text.

Table 44 Conversational Agent application technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-ConvA-01	The agent shall support more than one language.	The agent shall support English and Greek.	FUNC-ConvA-08 FUNC-ConvA-09
TECH-ConvA-02	The agent shall respond to all user input if there is internet connection.	The agent shall respond to input related to the ALAMEDA project and its pilots. For all other input the agent shall return a default predefined response.	FUNC-ConvA-01 FUNC-ConvA-02 FUNC-ConvA-03 FUNC-ConvA-05
TECH-ConvA-03	The agent shall be in constant communication with the Chatbot Android App.	The connection between the agent and the android app shall not be interrupted when the user's smartphone is connected to the internet.	FUNC-ConvA-06 FUNC-ConvA-11
TECH-ConvA-04	The agent shall take data privacy into consideration.	The agent shall not store any of the user's data on the server.	FUNC-ConvA-07 FUNC-ConvA-10

Table 45 Conversational Agent application system specifications

Component Name / ID	Conversational Agent
Partner(s) name	UNIC
Short Description	The Conversational Agent will engage patients and caregivers in cognitive and medical data collection by carefully constructed human-like interactions.
Core Functions	- Engage patients/caregivers by simulating human-like conversations. - Collect data related to medical questionnaires and non-disease related factors. - Collect data about the patients' emotional well-being.
Related System Requirements	RAM > 4GB A machine that can run Docker
Related ALAMEDA Tasks	T4.2, T4.3
APIs	Not ready yet
Inputs	User text input

Component Name / ID	Conversational Agent
Outputs	Text
Operating Systems	Windows and Linux
Programming languages & frameworks:	- Python - Rasa - spaCy - nltk - Tensorflow
Exposed services	API endpoints
User interface	No user interface
Database and databases tools	N/A
Hardware dependencies	N/A
Hardware devices involved in any use cases the component applies to	Servers Android smartphones

5.1.3.3 *Chatbot Android Application*

The Chatbot Android Application will provide a user-friendly graphical interface for patients to interact with the Conversational Agent. Additionally, the application will enclose a non-intrusive notification system for sending reminders to the users, for instance in case a patient forgot to complete their recurring medical questionnaires.

Table 46 Chatbot Android application technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-ChatApp-01	The application shall support more than one language.	The application shall support English and Greek.	FUNC-ChatApp-06 FUNC-ChatApp-07
TECH-ChatApp-02	The application shall be available for Android devices.	The application should work on Android phones running on SDK Android 8.0 (API level 26) and above.	FUNC-ChatApp-08
TECH-ChatApp-03	The module shall establish a communication channel with other ALAMEDA components.	The module shall share the gathered data in an appropriate and secure way (i.e., REST APIs).	FUNC-ChatApp-01 FUNC-ChatApp-02 FUNC-ChatApp-03 FUNC-ChatApp-04

Table 47 Chatbot Android application system specifications

Component Name / ID	Chatbot Android Application
Partner(s) name	UNIC

Short Description	The Chatbot Android Application will provide a user-friendly graphical interface for patients to interact with the Conversational Agent.
Core Functions	- Provide a graphical interface for interacting with the agent - Sending non-intrusive notifications and reminders
Related System Requirements	SDK Android 8.0 (API level 26) and above
Related ALAMEDA Tasks	T4.2, T4.3
APIs	N/A
Inputs	User text input
Outputs	Text from the server hosting the conversational agent
Operating Systems	Android OS
Programming languages & frameworks:	Java & Kotlin
Exposed services	API endpoints
User interface	Android Application
Database and databases tools	Room persistence library (an abstraction layer over SQLite)
Hardware dependencies	N/A
Hardware devices involved in any use cases the component applies to	Android Smartphones

5.1.3.4 *Mood Estimation Android Application (MEAA)*

MEAA is responsible for estimating the mood of the patient through two main functionalities: (a) An activity where the patient will be able to evaluate his/her day; (b) A daily questionnaire. While the patient is responding to the questionnaire, an emotion recognition algorithm will run on the background in order to capture the general feeling of the patient and shape a more holistic view about his/her mood.

Table 48 MEAA technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-MEAA-01	The MEAA module should have a clear view of the face of the patient.	The MEAA module should be installed and run on a smartphone with a front camera of at least 30fps.	FUNC-MEAA-01 FUNC-MEAA-02

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-MEAA-02	The MEAA module shall shape a view on the patient's mood through different kinds of inputs.	The MEAA module shall be able to store answers of the patients for further analysis and acquire data from available Android APIs related to the sensors and the interaction of a user with his/her phone.	FUNC-MEAA-03 FUNC-MEAA-04 FUNC-MEAA-05
TECH-MEAA-03	The MEAA module shall establish a communication channel with the rest of the ALAMEDA components.	The MEAA module shall share the gathered data in an appropriate and secure way (i.e., REST APIs, publish/subscribe models etc.).	FUNC-MEAA-06
TECH-MEAA-04	The MEAA module shall provide a good degree of usability.	The MEAA module shall present a very simple interface without including unnecessary information and elements.	FUNC-MEAA-07
TECH-MEAA-05	The MEAA module shall take privacy and personal information into consideration,	The MEAA module shall present a clear privacy policy to the patients, explaining in detail the data that will be gathered.	FUNC-MEAA-08

Table 49 MEAA system specifications

Component Name / ID	Mood estimation Android Application (MEAA)
Partner(s) name	CTL
Short Description	MEAA is responsible for estimating the mood of the patient through two main functionalities: (a) An activity where the patient will be able to evaluate his/her day; (b) A daily questionnaire. While the patient is responding to the questionnaire, an emotion recognition algorithm will run on the background in order to capture the general feeling of the patient and shape a more holistic view about his/her mood.

Core Functions	• An activity where the patient will be able to evaluate his/her day. • A daily questionnaire that the patient will fill. • An emotion recognition detection algorithm.
Related System Requirements	Android 10 (API 29) and above
Related ALAMEDA Tasks	T4.4
APIs	N/A
Inputs	• User input while filling the evaluation of the day. • User input while filling the daily questionnaire.
Outputs	A JSON file which will contain the answers given by the patient to both the aforementioned activities, the result of the emotion recognition detection and statistics about the applications with which the patient interacted the most during the day
Operating Systems	Android OS
Programming languages & frameworks:	Java & Kotlin
Exposed services	JSON
User interface	The application's interface
Database and databases tools	SQLite
Hardware dependencies	-
Hardware devices involved in any use cases the component applies to	Android Smartphone

5.1.3.5 *Line Tracking Test application*

The Line Tracking Test is an app designed to assess older adults' hand dexterity. It runs on tablet devices with Android® and a 10-inch tablet is recommended for its administration. Developed within the NoTremor EU project, the Line Tracking Test measures the ability to follow a randomly moving target (the cyan line) while ignoring the distracting target (the red line). The Line Tracking Test can identify different components of the human movement (e.g., reaction time, movement time, and several internal time delays)

Table 50 Line Tracking Test technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-LTT-01	LTT should accurately capture the user's performance and accuracy.	The LTT should be installed in a 10-inch Android tablet.	FUNC-LTT-01 FUNC-LTT-02 FUNC-LTT-06

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-LTT-02	LTT should provide a high degree of usability.	LTT shall present a simple interface and provide appropriate, easy to understand instructions and feedback to the user.	FUNC-LTT-05 FUNC-LTT-07
TECH-LTT-03	LTT should be able to communicate with the rest of the ALAMEDA components.	LTT must share the gathered data in an appropriate and secure way (i.e., REST APIs, publish/subscribe models etc.).	FUNC-LTT-03 FUNC-LTT-04
TECH-LTT-04	LTT should ensure sufficient privacy and security.	LTT shall provide a secure mechanism for storing and retrieving data.	FUNC-LTT-08

Table 51 Line Tracking Test system specifications

Component Name / ID	Line Tracking Test / LTT
Partner(s) name	CERTH
Short Description	The LTT will be used by patients participating in the ALAMEDA Parkinson's pilots. Participants will use the VST once in each clinic visit (month1, month 3 and month 6 of the pilots). LTT performance data will be used to assess patient's hand movements.
Core Functions	The LTT allows for the detailed assessment of the hand dexterity of the user through a specially designed task
Related System Requirements	10-inch Android tablet
Related ALAMEDA Tasks	T2.2, T2.3, T4.1, T4.2, T5.4, T6.2, T6.6
APIs	Specific API addresses are not yet decided
Inputs	User's personal information and their interaction with the application
Outputs	LTT performance data at runtime and collected data for each trial
Operating Systems	Android
Programming languages & frameworks:	Angular, Cordoba, Hypertext pre-processor (PHP)
Exposed services	N/A
User interface	The application's interface (main menu and testing interface)
Database and databases tools	N/A
Hardware dependencies	10-inch tablet
Hardware devices involved in any use cases the component applies to	• Servers • Tablets

5.1.3.6 *Virtual Keyboard application*

The Virtual Keyboard is designed to assess users typing patterns on a mobile device. It runs on mobile devices with Android®. It features a custom software keyboard, similar to the Android OS default keyboard, including all modern functionalities, such as word prediction and auto-correction which replaces the Android OS default keyboard. The software behind the keyboard captures keystroke-related data (key pixel coordinates and timestamps of key presses and releases), as well as typing metadata, i.e., number of deletes, number of characters typed, typing session duration, deliberate long-press events, and the application where the user typed, while the content of the typed text is not recorded.

The main application includes the Informatics and Telematics Institute (ITI) keyboard as a library. The users are prompted to choose the provided keyboard as the default system keyboard. Whenever the keyboard is used, the library saves keyboard sessions within an SQLite database inside the phone. Functions are provided to retrieve non-sent data from the database, as well as to set those data as "sent" afterwards. Periodically, previously not send keyboard sessions are grouped within a json string and are sent to a dedicated server through Hypertext Transfer Protocol (HTTP) POST requests. Specific API addresses are not yet decided.

Table 52 Virtual Keyboard application technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-VK-01	VK should accurately capture user's typing activity.	VK should be installed in an Android smartphone.	FUNC-VK-01 FUNC-VK-02
TECH-VK-02	VK should provide a high degree of usability.	VK shall not interfere with the use of the mobile device.	FUNC-VK-03 FUNC-VK-04
TECH-VK-03	VK should be able to communicate with the rest of the ALAMEDA components.	VK must share the gathered data in an appropriate and secure way (i.e., REST APIs, publish/subscribe models etc.).	FUNC-VK-03 FUNC-VK-04
TECH-VK-04	VK should ensure sufficient privacy and security.	VK shall provide a secure mechanism for storing and retrieving data.	FUNC-VK-05 FUNC-VK-06

Table 53 Virtual Keyboard application system specifications

Component Name / ID	Virtual Keyboard (VK)
Partner(s) name	CERTH
Short Description	The keyboard will be used by participants in the ALAMEDA Multiple Sclerosis, Parkinson's & Stroke pilots. Participants will use the keyboard in their everyday life. The data being collected will be used

	for correlation and statistical analysis, as well as for the development of machine learning models, using as target/ground truth the provided questionnaires.
Core Functions	The ITI keyboard is functioning as any other mobile keyboard. It is appearing whenever there is the need to input text within a field. While writing the keyboard collects key press/release timing related information.
Related System Requirements	Smartphone with android OS
Related ALAMEDA Tasks	T2.2, T2.3, T4.1, T4.2, T5.4, T6.2, T6.3, T6.4, T6.6
APIs	Specific API addresses are not yet decided
Inputs	User interaction with the mobile keyboard
Outputs	Timing related information about key presses/releases
Operating Systems	Android
Programming languages & frameworks:	Android, Java, C++
Exposed services	N/A
User interface	The keyboard is imported as a library and contains only the UI consisting of the keyboard, in multiple languages.
Database and databases tools	SQLite for the storage of keyboard sessions
Hardware dependencies	Mobile phone with Android
Hardware devices involved in any use cases the component applies to	• Servers • Mobiles phones • Network (WiFi, Mobile)

5.1.3.7 *Virtual Supermarket Test*

The Virtual Supermarket Test is an app designed to assess older adults' cognition through a simple task modelled on an everyday activity. It runs on tablet devices with Android® operating system and a 10-inch tablet is recommended for its administration. The latest version of VST comprises a fully automated, self-administered screening routine that can be completed in the span of 30 minutes. An interactive training session ensures that users are familiarized with the operation of the tablet and the various actions they would have to perform during testing. After completion of training a test session is administered three times through an automated administration routine. The VST is designed to mimic one of the most common activities of daily living, daily shopping in a supermarket. During the test sessions, a shopping list is provided to the user who is allowed to navigate freely, buy the products they are instructed to buy and proceed to pay at the till, by entering the correct amount. The application is aimed at activating a multitude of cognitive processes namely visual and verbal memory, executive function, attention, and spatial navigation with the emphasis placed on executive function. The need of simultaneous activation of different cognitive processes makes the program challenging enough to correspond to the ability of the target population while reducing ceiling effects. The latest version of the VST includes advanced navigation metrics with the virtual space divided into three zones (green, yellow, and red). Different zones represent different deviations from a pattern of optimal navigation for task completion. The diagnostic utility of the VST has been validated in different populations and it has also been validated against electroencephalography (EEG) biomarkers.

Table 54 Virtual Supermarket Test technical requirements

Requirement_ID	Description	Acceptance criteria	Related FUNC/NON-FUNC reqs
TECH-VST-01	VST should accurately capture user's performance in the virtual shopping task	VST should be installed in a 10-inch Android tablet	FUNC-VST-01 FUNC-VST-02
TECH-VST-02	VST should provide a high degree of usability	VST shall present a simple interface and provide appropriate, easy to understand instructions and feedback to the user	FUNC-VST-05 FUNC-VST-06 FUNC-VST-07 FUNC-VST-08
TECH-VST-03	VST should be able to communicate with the rest of the ALAMEDA components	VST must share the gathered data in an appropriate and secure way (i.e., REST APIs, publish/subscribe models etc.)	FUNC-VST-03 FUNC-VST-04
TECH-VST-04	VST should ensure sufficient privacy and security	VST shall provide a secure mechanism for storing and retrieving data	FUNC-VST-09

Table 55 Virtual Supermarket Test system specifications

Component Name / ID	Virtual Supermarket Test/ VST
Partner(s) name	CERTH
Short Description	The VST will be used by patients participating in the ALAMEDA pilots. Participants will use the VST once in each clinic visit (month1, month 3 and month 6 of the pilots). VST performance data will be used to assess patient's cognition.
Core Functions	The VST provides the ability to the user to simulate a classic procedure of touring and picking a list of products in a virtual supermarket and paying for these products choosing the right amount of money at the cashier.
Related System Requirements	10-inch Android tablet
Related ALAMEDA Tasks	T2.2, T2.3, T4.1, T4.2, T5.4, T6.2, T6.3, T6.4, T6.6
APIs	Specific API addresses are not yet decided
Inputs	User's personal information and their interaction with the application
Outputs	The application collects and uploads VST performance data at runtime and collected data for each trial.

Operating Systems	Android
Programming languages & frameworks:	Unity, C#
Exposed services	N/A
User interface	The application includes a main menu with extra tabs regarding options and the game tab where the user plays by tapping the related buttons.
Database and databases tools	N/A
Hardware dependencies	Tablet with Android
Hardware devices involved in any use cases the component applies to	• Servers • Tablets • Network (WiFi, Mobile)

5.2 AI Toolkit Architecture

Having presented the technical dimension of the AI Toolkit components above (see Subsection 5.1.1), we now present the architecture for the first iteration of the toolkit. The implementation of the AI Toolkit v1 is the objective of T5.3 and will be reported in the upcoming deliverable D5.3, which is due on M18. Figure 4 displays the overall architecture of v1 of the AI Toolkit.

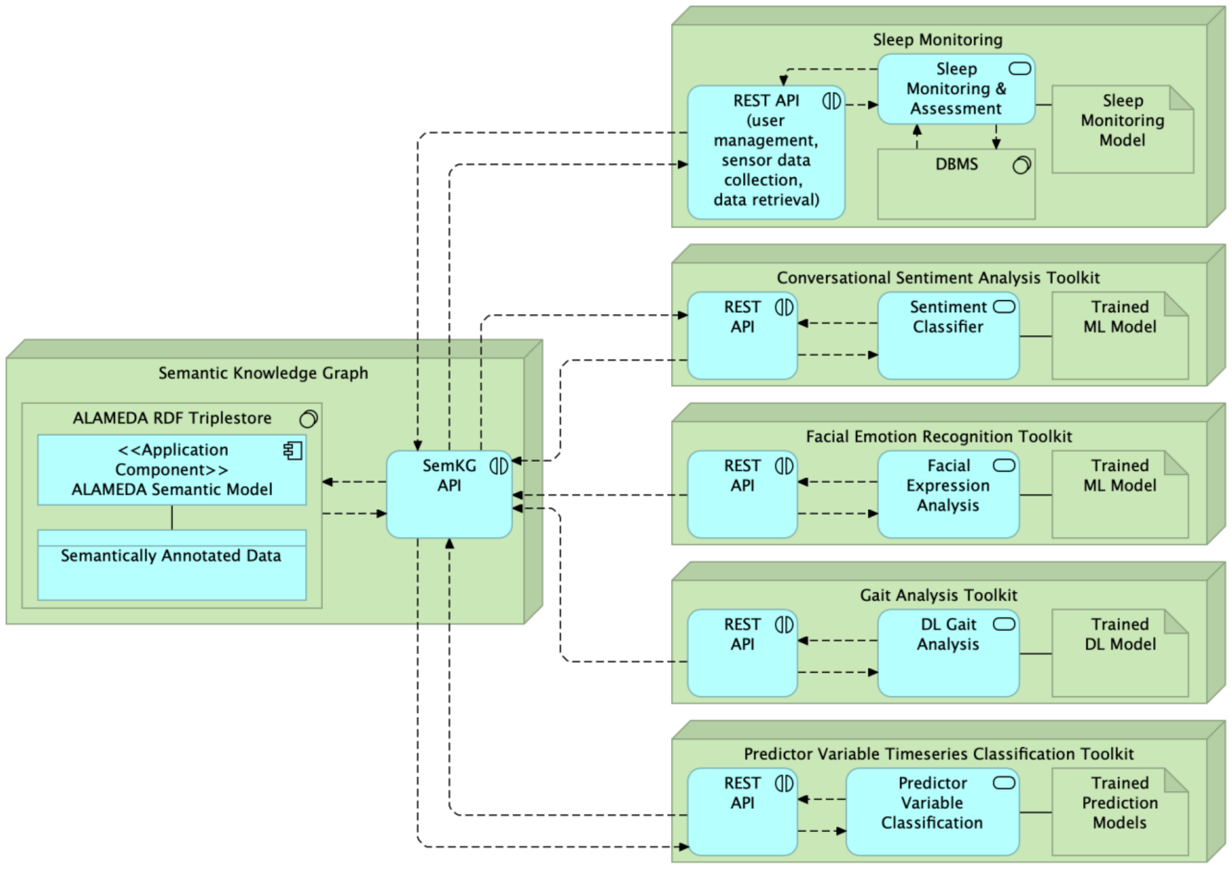

Sleep Monitoring
REST API (user management, sensor data collection, data retrieval)
Sleep Monitoring & Assessment
DBMS
Sleep Monitoring Model
Conversational Sentiment Analysis Toolkit
REST API
Sentiment Classifier
Trained ML Model
Semantic Knowledge Graph
ALAMEDA RDF Triplestore
<<Application Component>> ALAMEDA Semantic Model
Semantically Annotated Data
SemKG API
Facial Emotion Recognition Toolkit
REST API
Facial Expression Analysis
Trained ML Model
Gait Analysis Toolkit
REST API
DL Gait Analysis
Trained DL Model
Predictor Variable Timeseries Classification Toolkit
REST API
Predictor Variable Classification
Trained Prediction Models

Figure 4. Overall architecture of the first iteration of the AI Toolkit

The following design principles currently hold for the envisioned first iteration of the ALAMEDA AI Toolkit:

- As already discussed above, the SemKG component (Semantic Knowledge Graph) plays a central role, since it constitutes the primary means for establishing semantic interoperability among the rest of the components of the toolkit.
- Not all of the other components are currently expected to retrieve information from the SemKG; all of them, however, feed the latter with the outputs of their analyses.
- The other AI-powered components are all based on ML (some of them specifically on DL) and are accompanied by pre-trained models. Consequently, they can also be used independently by third parties, without the need to interact with the SemKG.
- The SemKG can also be used independently, since it will be available with pre-loaded anonymized data conforming to the semantic model (i.e., schema), which will also be available for external researchers and third-party stakeholders.

Based on the pilot outcomes, the design principles of the AI Toolkit will be reiterated and the second version of the architecture will be reported in D5.4 (due M20), while the AI Toolkit v2 will be presented in D5.5 (due M30).

5.3 Main ALAMEDA Architecture

The ALAMEDA AI Toolkit described above will be a central resource for either readily available AI/ML tools preconfigured and pre-trained, that can be directly used by researchers or also allowing for customization and training through user provided datasets for more fine-tuning to the more advanced cases and users along. It should demonstrate many out-of-the-box offered functionalities, while user configurability would make it more adaptable to more advanced use cases and for experimentation.

The AI Toolkit plays a central role in the ALAMEDA platform architecture that will be used to demonstrate ALAMEDA at the pilot sites. To this end, the AI Toolkit is surrounded with components and services that perform data collection and user applications that the ALAMEDA users will be able to use. The architecture presented in this section provides a high-level overview of the different layers of the ALAMEDA solutions, as well as the way the different components are situated within. More specifically, in the remainder of this section, the Logical and Technical architectures of ALAMEDA are presented, both of which have been designed based on the outcomes of the functional and non-functional requirements presented in the previous Section.

5.3.1 ALAMEDA Architecture Guiding Principles

Architecture principles are essentially a group of basic rules that guide the architectural solution, as well as the evolution, of the ALAMEDA platform. They are meant for driving the decisions the project makes in terms of technical activities, while at the same time they can by no means be amended or abandoned.

The guiding principles adopted so far in the project are presented in Table 56.

Table 56 ALAMEDA architecture guiding principles

Principle ID		
ARCH-PRINC-01	**Full legal compliance**	Every Information system must be designed and implemented in adherence with all applicable legislation, not only with respect to information systems' principles but additionally with the specific domain legislation it aims to serve. Moreover, as the legislation evolves, so should the system be able to quickly adapt.
ARCH-PRINC-02	**Interoperability**	The solution designed must not be a closed ecosystem; it must demonstrate to some degree interoperability with external systems. The European Interoperability Framework can be used as a guideline, along with specific requirements imposed by the project's objectives.
ARCH-PRINC-03	**Technology independence of**	The ALAMEDA platform application components must be independent of the underlying technologies

	applications	on which they are consumed. Despite the diversity in technologies and frameworks, ALAMEDA components must be treated as black boxes, with strictly defined interfaces based on commonly accepted standards, while at the same time they are packaged and deployed in a format that is independent of the underlying environment on which they run. Effectively, application independence allows for increased cost effectiveness and time management with respect to the development, upgrade, and operation of said applications.
ARCH-PRINC-04	**Scalability by design**	The ALAMEDA application components should be able to scale independently by adding more instances in a linear manner.
ARCH-PRINC-05	**Service orientation**	Application components in ALAMEDA must be regarded as services, each loosely coupled with the others, implementing a bounded functionality context. To realise interoperability, transparency and hassle-free flows among the application services, strong governance and open standards are required.
ARCH-PRINC-06	**Common Vocabulary and Data Definition**	The core data handled by the ALAMEDA platform need to be clearly defined, based on open definitions and standards as much as possible and then provide extensions, to address interoperability on the data level.

5.3.2 ALAMEDA architecture decisions

ALAMEDA is a complex project that requires several different components and services to interoperate and work as a whole. Due to the heterogeneity in the technologies of the different components, as well as the data transfer flow needs among them, several architectural decisions need to be decided among the consortium members, be documented, and communicated. The architectural decisions arrive usually through discussions among technical partners, in or during technical meetings and aim to solve complex issues that the platform needs to tackle. Typically, these complex issues affect the platform as a whole and are not simple bilateral issues between two components.

Typically, an architecture decision includes both the context of why this decision was made, as well as its potential impact. At the time of writing deliverable D5.1, and with tasks that deal with platform security and integration just starting, the main decision taken is presented in Table 57.

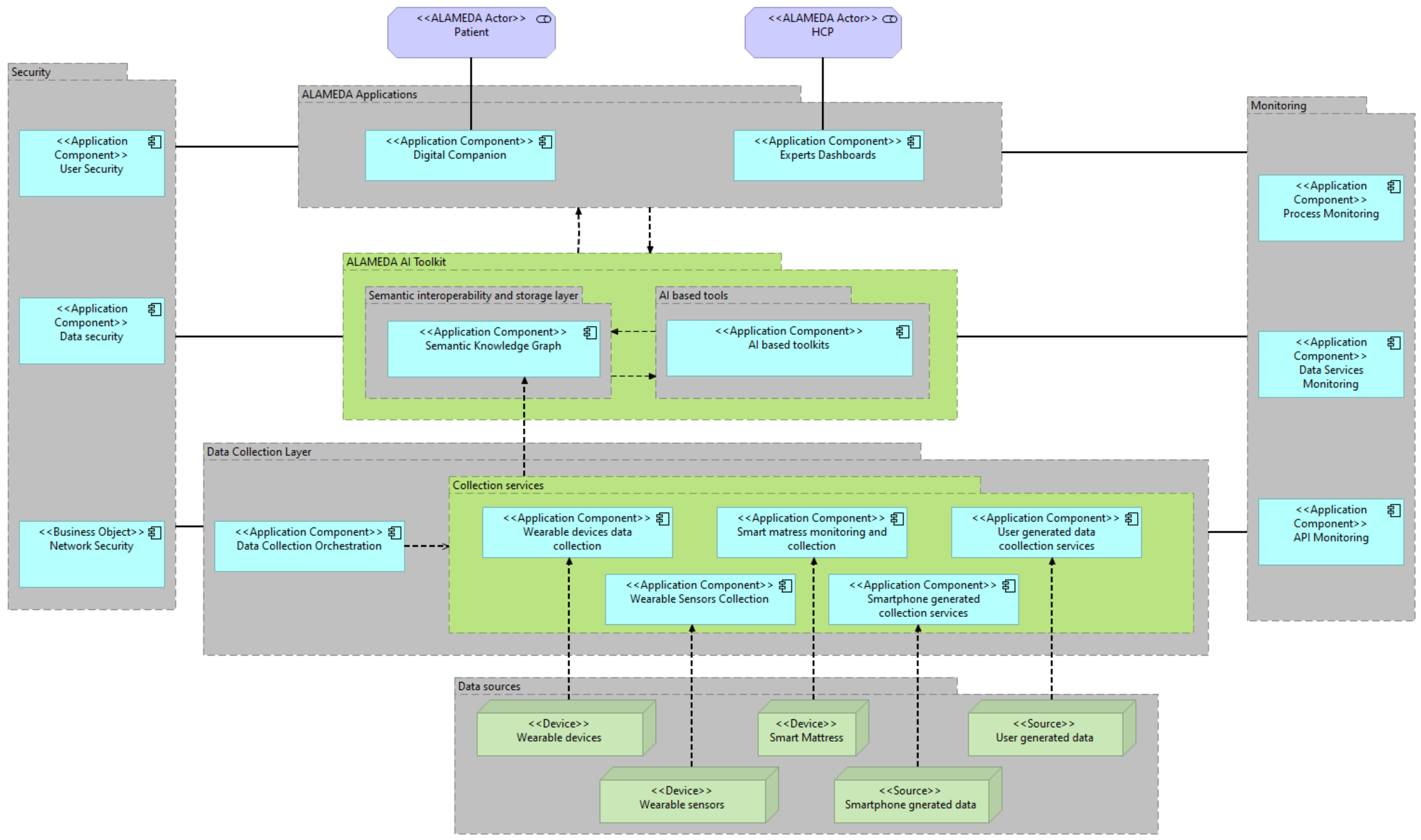
Table 57 Architecture decision AI-ARCH-DEC-01

Decision ID	AI-ARCH-DEC-01
Architectural decision	Implementation of a Semantic Knowledge Graph (SemKG)
Domain	AI Toolkit architecture
Issue	The semantic models developed within T4.1 constitute the primary semantic interoperability facilitator for the various AI Toolkit components. However, the latter are not compatible with the semantic model and ontology technologies and are producing data in formats other than RDF (JSON, text, csv), while at the same time they need to store data in and retrieve data from the ontology.
Assumptions	Transformation between components' produced outputs to RDF must be performed and vice versa.
Motivation	The populated ontology holds all the data generated both by the data collection layer and the AI Toolkit, while they need to be consumed by the Digital Companion and the Experts' Dashboards.
Alternatives	-
Decision	A transformation layer will be built in the form of a "semantic knowledge graph" and will be integrated in the SemKG module (see Sections **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** and **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**) to transform JSON to RDF format and vice versa, for the facilitation of the storage of data in and the retrieval of data from the ontology, both for development and demonstration purposes.
Justification	There needs to be a central point in the ALAMEDA platform where this transformation takes place, evolving in parallel with the ontology definitions. It would otherwise require great effort to recreate the same exact functionality in many different components, leading also to lots of error-prone custom code and redundancy.
Implications	-
Related decisions	-

Additional architectural decisions are bound to be taken as the technical work in ALAMEDA evolves and all tasks are starting in the next months. We anticipate that at least two such decisions, namely in the domains of user management and user authorisation and authentication across the platform, will be taken. These decisions, as well as others taken will be presented in deliverable D5.4 (M20).

5.3.3 Logical architecture

The purpose of the logical architecture is to provide an overview of the target solution at the logical level. It consists of all the identified logical application components, distributed, and presented in different technical layers. The following diagram presents the ALAMEDA logical architecture.



<<ALAMEDA Actor>> Patient
<<ALAMEDA Actor>> HCP
Security
<<Application Component>> User Security
<<Application Component>> Data security
<<Business Object>> Network Security
ALAMEDA Applications
<<Application Component>> Digital Companion
<<Application Component>> Experts Dashboards
Monitoring
<<Application Component>> Process Monitoring
<<Application Component>> Data Services Monitoring
<<Application Component>> API Monitoring
ALAMEDA AI Toolkit
Semantic interoperability and storage layer
<<Application Component>> Semantic Knowledge Graph
AI based tools
<<Application Component>> AI based toolkits
Data Collection Layer
<<Application Component>> Data Collection Orchestration
Collection services
<<Application Component>> Wearable devices data collection
<<Application Component>> Smart matress monitoring and collection
<<Application Component>> User generated data coollection services
<<Application Component>> Wearable Sensors Collection
<<Application Component>> Smartphone generated collection services
Data sources
<<Device>> Wearable devices
<<Device>> Smart Mattress
<<Source>> User generated data
<<Device>> Wearable sensors
<<Source>> Smartphone gnerated data

Figure 5. ALAMEDA Logical Architecture diagram

The ALAMEDA Logical architecture is comprised of the following logical layers:

1) Data Collection and sources
2) AI Toolkit
3) ALAMEDA applications
4) Security layer
5) Monitoring layer

The data collection layer is responsible for collecting the data from various sources and bringing them into the system. These data sources include, among others, wearable device data, smart sensors data (insole, bracelets, mattresses), questionnaires, as well as images. A full list is available in Table 58.

Table 58 ALAMEDA logical data sources

Module	Description
<<Device>> **Wearable Devices**	These devices include the Fitbit wearables and smart bracelets, that can capture physical activity, heart rate, sleep, and other physiological data
<<Device>> **Wearable Sensors**	These devices include the smart belts and the insole sensors that can capture kinetic data
<<Device>> **Smart mattress**	These devices include smart mattresses that can monitor and capture sleep related data
<<Source>> **Smartphone generated data**	These data sources include all smartphone related data sources such as the camera and GPS, that can capture images and videos, as well as location.
<<Source>> **User generated data**	These data sources include user generated data, such as the responses to questionnaires (medical or application specific), outcomes of the line tracking test, the virtual supermarket etc, in general data generated by the user interacting with the ALAMEDA applications

The data collection layer services are responsible for collecting the data from the various sources presented in the previous and bring them into the context of the ALAMEDA platform (Table 59).

Table 59 Data collection services

Module	Description
<<Application Component>> **Wearable Devices Data Collection Service**	An application component that receives data from the wearable devices and forwards it to the Semantic interoperability and storage layer

<<Application Component>> **Wearable Sensors Data Collection Service**	An application component that receives data from the wearable sensors and forwards it to the Semantic interoperability and storage layer
<<Application Component>> **Smart mattress Data Collection Service**	An application component that receives data from the smart mattresses and forwards it to the Semantic interoperability and storage layer
<<Application Component>> Smartphone generated Data Collection Service	An application component that receives data from the smartphone devices and forwards it to the Semantic interoperability and storage layer
<<Application Component>> User generated Data Collection Service	An application component that receives user generated input and forwards it to the Semantic interoperability and storage layer. These services are typically parts of the ALAMEDA applications, namely the Digital Companion.

The AI Toolkit layer, also presented in section 5.2, is comprised of the Semantic Knowledge Graph and the AI-based Toolkits (Table 60).

Table 60 AI Toolkit layer

Module	Description
<<Application Component>> **Semantic Knowledge Graph**	An application component that through an exposed API, receives data from the data collections services, offers data to the AI based toolkits and additionally serves data to the ALAMEDA applications in JSON format. Includes a transformation layer to transform the JSON based data from the collection services to the RDF format of the ontology.
<<Application Component>> **AI Toolkits**	An application component that receives data from and/or submits data to the Semantic Knowledge Graph and performs AI-based assessments on available collected data. Assessment outcomes are then stored in SemKG.

The ALAMEDA applications layer contains the logical application components that the ALAMEDA users operate (Table 61).

Table 61 Applications layer

Module	Description
<<Application Component>> **Digital Companion**	An application component comprises the mobile application that the ALAMEDA patient users operate, which allows them to perform various tests, answer questionnaires, see statistics about their physical, nutritional, and emotional status etc.
<<Application Component>> **Experts' Dashboards**	An application component that provides a web interface with administrative dashboards that provide to the ALAMEDA health care professionals users targeted information on their patients' status (both collected and AI based assessed data)

The Security layer is a cross-layer that handles the security services for all other layers of the platform. Specifics on the security aspects of ALAMEDA will be explored in the context of T4.5.

Table 62 Security Layer

Module	Description
<<Application Component>> **User Security**	An application component that allows the verification of user identity and grants them access to the ALAMEDA system.
<<Application Component>> **Data Security**	An application component that provides authorized access to the data of the ALAMEDA system.
<<Business Object>> **Network Security**	A set of tools and protocols that enable secure network access to the ALAMEDA system.

Monitoring is a cross-layer in charge of the monitoring services for all other layers. It supervises the system's performance through monitoring the services found in all the layers of the ALAMEDA system. Monitoring, as well as other integration factors that may directly affect the ALAMEDA logical architecture will be explored in T5.4.

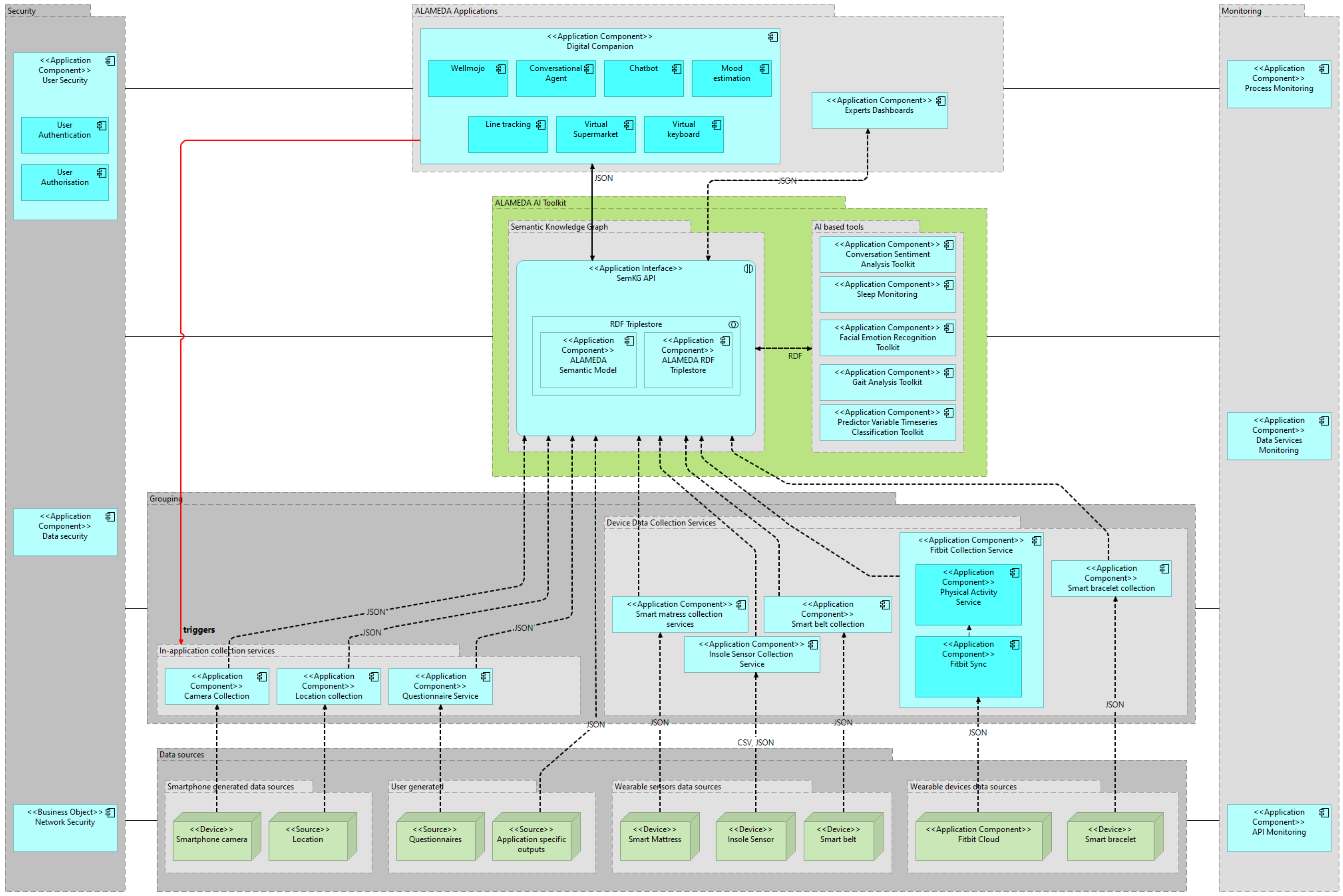
Table 63 Monitoring Layer

Module	Description
<<Application Component>> **Process Monitoring**	An application component that provides a web service that allows the monitoring of behavior of process realized by

	the ALAMEDA platform.
<<Application Component>> **Data Services Monitoring**	An application component that provides web service that allows the monitoring of availability and performance of the Data layer services.
<<Application Component>> **API Monitoring**	An application component that allows the monitoring of availability and performance of services of the SemKG API and other services APIs.

5.3.4 ALAMEDA Technical architecture

The Technical Application Architecture builds upon the Logical Application Architecture by selecting the components that will take part in the final solution as well as defining the associations between these components. More specifically, the technical application architecture serves to provide a view of the target application architecture at the technical level which consists of different technical application components.



Security
<<Application Component>> User Security
User Authentication
User Authorisation
<<Application Component>> Data security
<<Business Object>> Network Security
ALAMEDA Applications
<<Application Component>> Digital Companion
Wellmojo
Conversational Agent
Chatbot
Mood estimation
Line tracking
Virtual Supermarket
Virtual keyboard
<<Application Component>> Experts Dashboards
JSON
JSON
Monitoring
<<Application Component>> Process Monitoring
<<Application Component>> Data Services Monitoring
<<Application Component>> API Monitoring
ALAMEDA AI Toolkit
Semantic Knowledge Graph
<<Application Interface>> SemKG API
RDF Triplestore
<<Application Component>> ALAMEDA Semantic Model
<<Application Component>> ALAMEDA RDF Triplestore
RDF
AI based tools
<<Application Component>> Conversation Sentiment Analysis Toolkit
<<Application Component>> Sleep Monitoring
<<Application Component>> Facial Emotion Recognition Toolkit
<<Application Component>> Gait Analysis Toolkit
<<Application Component>> Predictor Variable Timeseries Classification Toolkit
Grouping
triggers
JSON
JSON
JSON
In-application collection services
<<Application Component>> Camera Collection
<<Application Component>> Location collection
<<Application Component>> Questionnaire Service
Device Data Collection Services
<<Application Component>> Smart matress collection services
<<Application Component>> Smart belt collection
<<Application Component>> Insole Sensor Collection Service
<<Application Component>> Fitbit Collection Service
<<Application Component>> Physical Activity Service
<<Application Component>> Fitbit Sync
<<Application Component>> Smart bracelet collection
JSON
JSON
JSON
CSV, JSON
JSON
JSON
Data sources
Smartphone generated data sources
<<Device>> Smartphone camera
<<Source>> Location
User generated
<<Source>> Questionnaires
<<Source>> Application specific outputs
Wearable sensors data sources
<<Device>> Smart Mattress
<<Device>> Insole Sensor
<<Device>> Smart belt
Wearable devices data sources
<<Application Component>> Fitbit Cloud
<<Device>> Smart bracelet

Figure 6. ALAMEDA Technical architecture

The following table contains the mapping of logical components on the technical components:

Logical Module	Technical Module(s)
Data sources	
<<Device>> **Wearable Devices**	- Fitbit Cloud API - Smart bracelet
<<Device>> **Wearable Sensors**	- Insole sensor - Smart belt
<<Device>> **Smart mattress**	- Smart mattress
<<Source >> **Smartphone generated data**	- Camera captured images and videos - User location
<<Source >> **User generated data**	- Questionnaires - Application specific metrics and outputs
Data collection services	
<<Application Component>> **Wearable Devices Data Collection Service**	- Fitbit collections Service (Fitbit sync and Physical activity services) - Smart bracelet collection service
<<Application Component>> **Wearable Sensors Data Collection Service**	- Insole sensor collection service - Smart belt collection service
<<Application Component>> **Smart mattress Data Collection Service**	- Smart mattress collection service
<<Application Component>> **Smartphone generated Data Collection Service**	- Camera collection via the Mood Estimation application - Location via Wellmojo
<<Application Component>> **User generated Data Collection Service**	- Questionnaire service - Collection of data via sub-applications of the Digital Companion
AI Toolkit	
<<Application Component>> **Semantic Knowledge Graph**	- SemKG API - RDF Triplestore - ALAMEDA Semantic Model
<<Application Component>> **AI Toolkits**	- Conversational Sentiment analysis toolkit - Sleep monitoring and assessment - Facial emotion recognition toolkit - GA toolkit

	- Predictor Variable Timeseries - Classification toolkit
ALAMEDA Applications	
<<Application Component>> **Digital Companion**	- Wellmojo - Conversational agent - Chatbot - Mood estimation application - Line tracking test - Virtual supermarket - Virtual keyboard
<<Application Component>> **Experts' Dashboards**	- Experts' dashboards (web application)
Security	
<<Application Component>> **User Security**	- User authentication and authorisation, user management services
<<Application Component>> **Data Security**	To be decided
<<Business Object>> **Network Security**	To be decided
Monitoring	
<<Application Component>> **Process Monitoring**	To be decided
<<Application Component>> **Data Services Monitoring**	To be decided
<<Application Component>> **API Monitoring**	To be decided

5.3.5 ALAMEDA Integration and Deployment Architecture

As described in the previous sections, the ALAMEDA architecture is composed of multiple components which should be effectively and smoothly integrated to provide the aspired functionalities and features, as documented in the ALAMEDA architecture and components specifications, to the stakeholders of ALAMEDA. Hence, the ALAMEDA platform will leverage the benefits of multiple open-source technologies, frameworks and tools that should be effectively combined to successfully implement and deliver the ALAMEDA platform which will address the elicited technical requirements described in Section 3 of the current deliverable.

As with all modern software products, the development process that will be adopted is based on an iterative incremental process in which multiple iterations are performed in order to produce incremental versions of the designed components. The ultimate goal of each iteration is to produce an incremental version of the ALAMEDA platform in which the various incremental versions of the components are effectively and smoothly integrated, verified and validated. The aforementioned process is referenced as an integration cycle of the development phase of the ALAMEDA platform. Each successfully completed integration cycle produces the new incremental version of the ALAMEDA platform, which is used as the basis for the generation of the next incremental of version of the version of the ALAMEDA platform towards the delivery of the final and fully functional version of the ALAMEDA platform. However, the described process dictates the requirement of a solid and robust integration process, which will safeguard and ensure the produced software quality of the implemented software artefacts, but will also guide the implementation and integration activities during the whole development phase of the ALAMEDA platform. In particular, the integration process will clearly define how and when the various components will be implemented and delivered by the various development teams of the different involved organizations and most importantly how these components will be effectively combined and smoothly integrated.

To this end, the ALAMEDA integration process will be based on the following main pillars:

• **A selection of mature and well-established open-source integration tools and techniques**, which enable and guarantee the proper planning, design and collaboration of the involved development teams. The list of tools and techniques includes:

- source code versioning and management tools,
- automated building tools depending on the utilized programming language,
- unit and integration testing frameworks,
- issue tracking,
- source code packaging and delivery tools,
- continuous integration and continuous delivery (CI/CD) tools,
- source code quality testing with code analysis tools.

• **A solid integration strategy**, which clearly and strictly defines the planning, designing and delivery of the various components utilizing the aforementioned integration tools and techniques for the smooth collaboration of the involved development teams

• **A robust integration plan**, which defines the integration activities that will be performed during each integration cycle and each incremental version of the ALAMEDA platform. The integration plan will take into consideration the dependencies between the various components by building a components dependency matrix, as well as the aspired releases of the platform in accordance with the ALAMEDA Description of Action.

- **A solid integration testing process**, which will dictate the validation and verification activities that will be performed on each integration cycle in the form of unit and integration tests, as well as through software quality testing with code analysis tools.

It should be noted that the complete and detailed documentation of the ALAMEDA integration process will be documented in the context of deliverable D5.3 on M18 as per the ALAMEDA Description of Action.

6. Data Management and Security

6.1Security Guidelines for Software Development

Source code quality testing with code analysis tools mentioned in Section **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** is expected to embed security tests. As a general rule, all code developed in the context of ALAMEDA should undergo security scrutiny by an automatic static code analyzer, that looks for usage of components with known vulnerabilities. For instance, one such tool is Snyk[11].

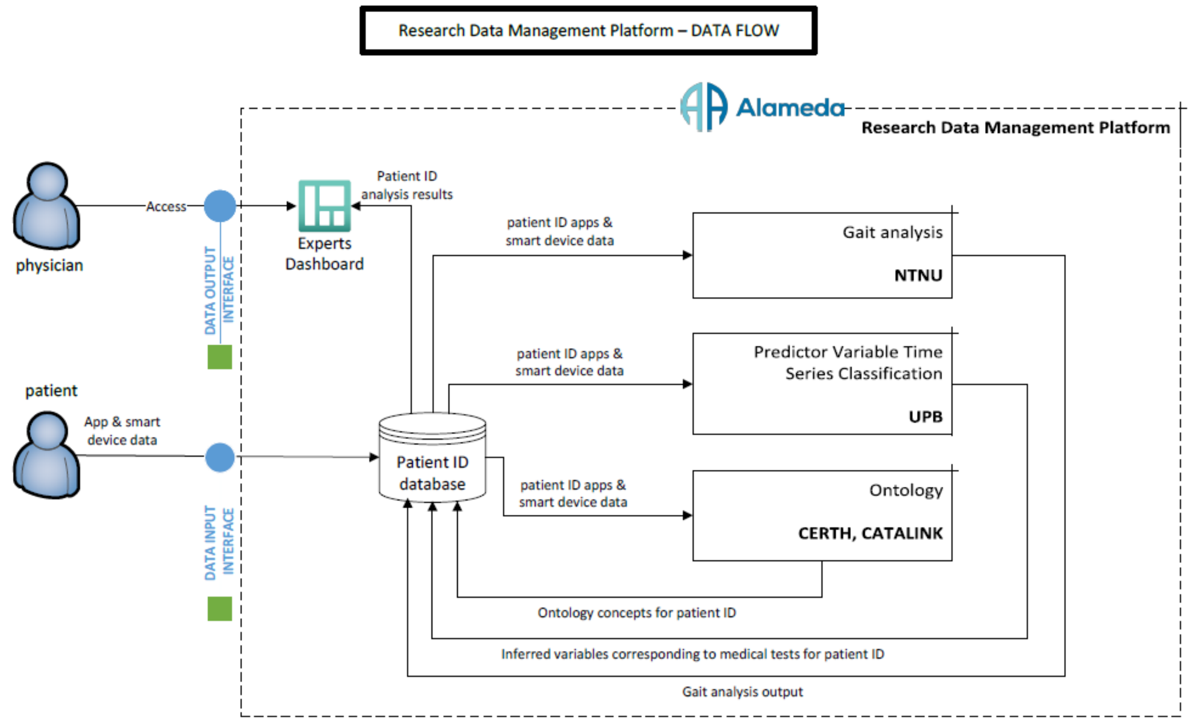

Research Data Management Platform – DATA FLOW
Alameda
Research Data Management Platform
physician
Access
DATA OUTPUT INTERFACE
Experts Dashboard
Patient ID analysis results
patient ID apps & smart device data
Gait analysis
NTNU
Predictor Variable Time Series Classification
UPB
patient
App & smart device data
DATA INPUT INTERFACE
Patient ID database
Ontology
CERTH, CATALINK
Ontology concepts for patient ID
Inferred variables corresponding to medical tests for patient ID
Gait analysis output

Figure 7. AI Toolkit components within the core Research Data Management Platform (RDMP)

As shown in Figure 7, we have currently identified three AI Toolkit components at the core of the RDMP:

- **Semantic Knowledge Graph** by CERTH/CTL
- **Predictor Variable Time Series Classification** by UPB
- **Gait Analysis (GA)** by NTNU

This is because all such systems need to access all data collected by apps & smart devices in ALAMEDA and need to learn from patient's data.

[11] https://snyk.io

All software code related to such components should be deployed inside the machines that compose the core Research Data Management Platform. Such machines may acquire, process and output data that is available only in the context of the core RDMP (e.g., dedicated buckets). In this way, researchers do not directly access data but run their algorithms on data and visualize the results/outputs. Network traffic exchanged by such machines may be logged using tools like Zeek[12] so that suspicious or unexpected network connections (that may exfiltrate data/represent a security threat) can be recorded/detected.

Conversely, the Facial expression classifier component of the AI Toolkit is expected to run directly on the patient's smartphone using a pre-trained model, inside the Mood Estimation Android App by CTL as it only needs to classify in real-time images from the smartphone camera. Its output in ALAMEDA is effectively treated as a feature for higher level classifiers within the core RDMP.

Detailed security specifications will be provided in the context of Task 4.5, based on a sound threat model that considers all components of the ALAMEDA Research Data Management Platform.

In the following, for each module of the AI toolkit, we briefly outline all software dependencies. This list is useful as "software asset inventory" to be able to quickly identify whether the AI toolkit is using components with known vulnerabilities or that may constitute a threat for data collection if not configured properly.

6.1.1 Conversational Sentiment Analysis toolkit libraries

The libraries used for the conversational sentiment analysis model are the following:

Flask, a micro web framework written in Python (https://flask.palletsprojects.com/en/2.0.x/)

NumPy, an open source project aiming to enable numerical computing with Python (https://numpy.org/)

Nltk, a suite of libraries written in Python for symbolic and statistical natural language processing for English (https://www.nltk.org)

Uwsgi, an application server for developing a full stack for building hosting services (https://uwsgi-docs.readthedocs.io/en/latest/)

Keras, an open-source software library that provides a Python interface for ANNs. Keras acts as an interface for the TensorFlow library (https://keras.io/)

TensorFlow, an end-to-end open-source platform to help you develop and train ML models (https://www.tensorflow.org/)

6.1.2 Mood Estimation Applications

NTNU uses Python as the core programming language and builds on the following:

[12] https://zeek.org

Keras, an open-source software library that provides a Python interface for ANNs. Keras acts as an interface for the TensorFlow library (https://keras.io/)

TensorFlow, an end-to-end open-source platform to help you develop and train ML models (https://www.tensorflow.org/)

PyTorch, an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (https://pytorch.org/)

The Mood Estimation Application developed by CTL is an Android application fully developed in Android Studio exploiting both Kotlin and Java language and the full set of functionalities and libraries that they provide. For the face detection and the facial landmarks extraction the application exploits ML KIT[13] which is a mobile SDK designed and developed by Google, aiming at bringing real-time machine learning capabilities to the smartphones. Google's ML Kit provides a vision API which, among others, includes a Face Detector, exploited within the frame of the application for monitoring the expressions of the patient. With respect to the rest of application functionalities, we also exploit the UsageStats API which gives us the possibility to gather information on the daily usage of the several applications within a smartphone and some open-source libraries, such as the "material-calendarview"[14] for displaying a fully functionable and formattable calendar to keep a history of the user's data. Other libraries include the JSch[15] for connecting the application to an external server, the MPAndroidChart[16] for displaying the charts and the GSON[17] library for saving the results as a JSON file.

6.1.3 Open tools and implementations

In [27], Walch et al. classify sleep stages among 31 people based on acceleration and heart rate signals collected via an Apple Watch. The obtained accuracy is 90% for sleep/wake classification and 72% for wake/ non-rapid eye movement (NREM)/rapid eye movement (REM) classification, when NNs are deployed. The respective code is publicly available on GitHub. Firstly, a script to access the data from the Apple Watch, written in Swift and Objective-C, is provided in [28]. The preprocessing steps that were followed, the algorithms used for the analysis and the classification tasks, as well as the code which produces the figures that illustrate the performance of the deployed models are included in the repository [29]. The ML pipeline is based on pre-built tools from scikit-learn in Python. A few MATLAB features and utilities are also present. Additionally, Walch shares a Python script that takes as input actigraphy and sleep diary data and infers possible points of sleep onset and offset [30].

Regarding PD, the mPower study [31] surveys 9520 volunteers through the mPower mobile application. It collects patient-reported demographic and clinical data and multimodal measurements while performing 4 activities (walking and standing, tapping, voice recording and memory assessment through a visuospatial game). The code for the mobile application is publicly available through a GitHub

[13] https://developers.google.com/ml-kit/
[14] https://github.com/prolificinteractive/material-calendarview
[15] http://www.jcraft.com/jsch/
[16] https://github.com/PhilJay/MPAndroidChart
[17] https://github.com/google/gson

repository of Sage-Bionetworks [32]. It is built for iOS, using the opensource Apple's ResearchKit framework [33] and the AppCore [34], which is a layer built on top of ResearchKit, forming the core of the five initial ResearchKit applications. Another available component is the BridgeSDK [35] for iOS which enables integration of other research studies applications with Sage Bionetworks' Bridge REST API. All the components are available via GitHub under specific licenses. Their implementation is based on the following programming languages: HTML, Cascading Style Sheets (CSS), Objective-C, Swift and Unix shell, among others. Finally, in this GitHub Repository [36], some Python and R scripts are provided as examples to access the data programmatically through Synapse. The code used for summary statistics and figures generation for the mPower publication [31] is also available. Similarly, some example scripts written in Python and R to access the Levodopa Response Study are given in [37].

Moreover, the authors in [38] present their DL approach that won in the 2017 PD Digital Biomarker Dialogue for Reverse Engineering and Assessments and Methods (DREAM) Challenge, based on the mPower data [31]. Their model detected PD based on acceleration signals collected with the mPower mobile application. The respective code is freely accessible via a GitHub repository [39]. The deployed programming languages span across Python, Perl and Unix shell commands while the main ML libraries used are scikit-learn, Theano, tensor and Lasagne.

Regarding the PD Digital Biomarker DREAM challenge, besides the open data used to train and test the algorithms, the deployed methods are described and the respective code is available through Synapse [39]. Full feature sets and the specific segmentation and training/test splits of the data used in the challenge are also provided along with the submitted implementations, so that the results could be reproducible and other researchers could further improve them. The respective scoreboards for each sub-challenge are also available. More information about this challenge and the winning methods can be found in [40].

Lonini shares their code (a Jupyter notebook) for the PD Digital Biomarker sub-challenge 2 in a GitHub repository [41]. They train a one-class Support-Vector Machine (SVM) with Radial Basis Function (RBF) kernel to estimate the severity of tremor, bradykinesia and dyskinesia based on both time and frequency domain features, following an anomaly detection approach. Moreover, Lonini et al. make use of the Clinician Input Study in PD (CIS-PD) dataset [42] to detect tremor and bradykinesia with CNNs and other statistical ensemble methods, based on signals collected via skin-mounted MC10 BioStampRC sensors [43]. The respective code is again maintained in a GitHub repository [44]. The implementation is in Python and Jupyter notebooks and the deployed libraries include TensorFlow, Keras, Theano and pandas.

Similarly, Shawen et al. [45] detect tremor and bradykinesia and estimate their severity with the help of random forests trained over acceleration data from an Apple Watch and a dorsal-mounted MC10 BioStampRC sensor (a subset of the CIS-PD dataset). The code regarding the data processing and analysis of this study is available on a GitHub repository [46]. Python and Jupyter notebooks are used to implement the proposed method while the scikit-learn library is deployed to build the respective ML models.

Moreover, Habets et al. [47] propose a logistic regression model to detect medication-state fluctuations among PD patients, based on both objective accelerometer and gyroscope data and subjective ecological momentary assessment (EMA) data. The code for the technical validation of this dataset, including merging and aligning these two data types, as well as for feature extraction and classification analysis is maintained on a GitHub repository [48]. The overall implementation is organized in a Jupyter Notebook and is written in Python. The scikit-learn library is used to build the proposed logistic regression model and its evaluation.

On the other hand, an open-source Python library for audio signal analysis is pyAudioAnalysis, uploaded to the Python Package Index (PyPI) repository [49]. It supports various audio-related functionalities, including feature extraction, classification, regression, supervised and unsupervised segmentation, as well as visualization. The source code is maintained on a GitHub repository [50] and further documentation can be found in [51]. This library has already been used to analyze speech recordings data from the mPower study and diagnose PD based on vocal features [52].

During the eNTERFACE of 2005, in the context of Project 2, several audio-video sequences in English were collected depicting various basic emotions [53]. Some C++ and MATLAB scripts for audio and video processing and emotion recognition are publicly available along with the respective data. Similarly, the Crowdsourced Emotional Multimodal Actors Dataset (CREMA-D) [54], that contains both video and audio (in English) recordings depicting different emotions is publicly available via a GitHub repository [55]. Some R scripts regarding the processing of these data are also available in this repository.

Moreover, Carnegie Mellon University (CMU) - Multimodal Corpus of Sentiment Intensity (MOSI) and Multimodal Opinion Sentiment and Emotion Intensity (MOSEI) dataset are two datasets which again contain both audio (in English) and video recordings and are publicly available via a GitHub repository [56]. In [57] the authors propose multi-attention RNNs to address the emotion recognition task. The respective code is also available through the previous GitHub repository [56]. The deployed programming language is Python and the DL models are built with the help of PyTorch.

Regarding Multiple Sclerosis (MS), firstly, in [58], gait-related features are extracted from people with Multiple Sclerosis (PwMS) and healthy controls via smartphone and smartwatches sensors, while performing Two-Minute Walk Tests (2MWT). These features are forwarded to logistic regression, SVM and random forests classifiers that detect MS. The MATLAB scripts for feature extraction are publicly available under a GNU General Public License through a GitHub repository [59], along with some data samples. A DL approach has also been proposed to address this problem based on the same dataset of IMU signals [60]. The respective training and testing code is provided in a Jupyter Notebook through a GitHub repository [61]. The script is written in Python and makes use of the TensorFlow, Keras and iNNvestigate libraries. Some MATLAB functions are also included.

Additionally, in [62], upper extremity features are extracted while PwMS and healthy controls are drawing shapes with the help of the mobile Floodlight POC application. These features can train regression models to estimate the 9-Hole Peg Test (9HPT) times. The MATLAB scripts used in this study to extract features are maintained in a GitHub repository [63], along with some data samples. Floodlight data are also available in another project maintained in GitHub [64] which detects MS with the help of decision trees, random forests, linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). The respective code is written in Python, using the scikit-learn library, and Tool Command Language (TCL), among other programming languages and is provided via the same GitHub repository.

Moreover, the implementation of the backend for Swiss MS Registry [65], which is a patient-centered research project for the documentation of the MS epidemiology and the quality of life of PwMS, is publicly available through a GitHub repository [66]. It is written in Python, using the Flask framework. There is also an open-source implementation of an Expanded Disability Status Scale (EDSS) scores calculator. Given the visual function, the brainstem function, the pyramidal function, the cerebellar function, the sensory function, the bowel and bladder function, the cerebral function and the ambulation scores, the final EDSS score is automatically calculated. The respective code is written in JavaScript and is maintained in a GitHub repository [67].

Regarding stroke, in a GitHub repository [68], signals from several wearable IMUs, embedded in a jacket, are collected to assess the range of motion and the arm rehabilitation of stroke patients. Along with the provided data, scripts, written mainly in MATLAB and C++, for the construction of the models are also publicly available. Additionally, in [69] grip force and load force signals are collected via a grasp rehabilitation device instrumented with force sensors from 15 subjects. The respective data and MATLAB scripts for signals processing and data analysis are publicly available via a GitHub repository [70].

Moreover, “Back on Track” is a mobile application that supports the remote upper-limbs rehabilitation of post-stroke patients through exercises selected from the Canadian Graded Repetitive Arm Supplementary Program. The patients are required to follow the proposed exercises through videos. After performing each exercise, the users’ effort level will be requested as feedback according to the adapted Borg scale. In this way, the physiotherapist will be able to monitor patients remotely, analyze their data and suggest personalized rehabilitation programs. The overall implementation is written in Java and is publicly available through a GitHub repository [71].

An exergame, based on Kinect, for balance rehabilitation of post-stroke patients has also been developed for a Bachelor thesis [72]. The respective open-source code is written in C# and is maintained in a GitHub repository [73]. The source code of another serious game for stroke rehabilitation is available through a GitHub repository [74]. It is built on the Unity platform game engine, in C#.

Finally, the LotsenApp is an application which collects data from stroke patients to create their digital patient files. It is developed as a component of the STROKE OWL project [75]. Its source code is available on a GitHub repository [76] and is based on the following programming languages: C#, TypeScript, HTML, Sassy CSS and JavaScript. Data security and encryption protocols play a vital role in this implementation. Another application suitable for stroke rehabilitation is BrainFit. It supports face detection for drooling measurement, reminders for medication dosage intake and exercises, as well as rehabilitation for speech impairment and hands weakness through games. It is built on Android studio and the source code is written in Java and PHP and is publicly available via a GitHub repository [77]. More information about this application can be found in [78], [79].

6.2 Available Datasets

An initial list of datasets of interest to ALAMEDA pilots has been compiled, focusing on data resembling the objectives envisioned in the pilots. Information on datasets for patient health metrics prediction based on wearables and self-reported data can be found in [80], while dataset indexes for FEA and emotion recognition based on speech can be found in [81] and [82], respectively.

6.2.1 Datasets Based on Wearable Sensors

Firstly, there are various datasets that contain motor measurements from wearable inertial sensors, pressure sensors or other biosensors. These datasets are mapped to the three pilots (10 PD-related datasets and 2 MS-related datasets so far).

6.2.1.1 *Parkinson's Disease*

The Levodopa Response Study [83], which is supported by the Michael J. Fox Foundation (MJFF), consists of two parts. In Part I, raw accelerometer data are collected from 28 PD patients with two smartwatches (a GeneActiv on the wrist of the most affected limb and a Pebble on the wrist of the least affected limb), as well as with a Samsung Galaxy Mini smartphone placed in a fanny pack, worn in front at the waist. Additionally, in PART II, raw accelerometer data are further collected from a subset of 17 PD patients with 5 Shimmer3 sensors, mounted on both upper limbs, both lower limbs and the waist. The examined types and placements of sensors are very convenient for the PD pilot in ALAMEDA, as in our case smartwatches, smart bracelets and smartphones will be deployed, as well and the prototype smart belt will extract waist motor features.

In the Levodopa Response Study, all the subjects are monitored in a 4-day period. The first and the last day monitoring takes place in clinic and the other two days at home. In the first visit, demographic and clinical data are collected, including patient-reported experiences and manifestations, Movement Disorder Society-Sponsored Revision of the Unified PD Rating Scale (MDS-UPDRS) I, II, II and IV scores. MDS-UPDRS III scores were extracted in the first and the last visit to the clinic for each PD patient. Additionally, all the PD patients performed some selected tasks in certain repetitions, such as walking activities, finger-to-nose tests and alternating hand movements among others, in clinical visits while they were asked to perform their regular activities when at home. The subset of patients who wore Shimmer3 sensors, were also requested to perform selected tasks in certain repetitions at their home, as well. For them, the individual item scores of the MDS-UPDRS are also provided for the last clinic visit. During the patients' monitoring at home, the subjects were asked to keep medication and sleep diaries and the patients who wore the Shimmer3 sensors also self-reported the severity of their symptoms while performing the requested tasks. Data are provided by the Synapse organization in a structured schema where each table corresponds to an entity or a set of variables. Besides these queryable tables, raw accelerometer data are provided in a txt format for each patient and each day of monitoring. All the data (68.5GB for Part I and 70.7 GB for Part II) will be fully available after following the 5-step access protocol of the Synapse organization.

The target variables correspond to tremor severity indices (0,1,2,3/4), the presence/absence of bradykinesia and dyskinesia for the total set of patients and additionally bradykinesia and dyskinesia severity indices (0,1,2,3/4) for the subset of patients who wore Shimmer3 sensors. The fact that MDS-UPDRS scores are provided for two visits enable the prediction of the deterioration or improvement, as well, in a short period of time. Consequently, the provided targets are very similar with what we want to predict in the ALAMEDA PD pilot. Moreover, self-reported symptoms and experiences could be considered as ground truth data. The Levodopa Response Study data may be used to train and test ML algorithms and build DL architectures, in advance, without having the ALAMEDA patients' data at our disposal. It is also possible that these data could be merged with the data acquired in the context of ALAMEDA. More information about this dataset could be retrieved from the publications [84], [85] for part I and II, respectively. Both studies have not been cited by other research papers that propose AI methods, yet. However, the respective binary and multiclass classification problems have already been addressed in the literature by many conventional ML and DL methods, including logistic regression, Naïve Bayes, k nearest neighbors (kNN) algorithms, decision trees, random forests and bagged trees, support vector machines, ANNs, like multilayer perceptrons (MLPs), CNNs, LSTMs and others.

The mPower Mobile PD Study [86] is a longitudinal study which surveys 9520 participants through the mPower mobile application. Firstly, the subjects were requested once to provide some demographic information (6805 participants responded from whom 1087 self-reported to face PD and 5581 do not face PD). They were also requested to provide on a monthly basis some self-reported MDS-UPDRS scores (0,1,2,3,4) (2024 participants responded) and PD Questinnaire-8 (PDQ-8) responses (5 scales) (1334 participants responded). Then they were asked to perform 4 activities up to 3 times per day (for PD patients: before and after taking their medication as well as at any other point of the day they wanted). These activities include walking and standing, tapping, voice recording and memory assessment through a visuospatial game. 3101 subjects participated in the walking and standing task, providing measurements from the smartphone inertial sensors (accelerometer, gyroscope and pedometer), which was placed in the subjects' pocket. 8003 subjects participated in the tapping activity, where tapping logs and accelerometer measurements were collected. Finally, 5826 and 968 subjects participated in the voice recording task and in the memory game, respectively. As each activity was performed more than once from many subjects, we have much more thousands of records at our disposal.

More information about this dataset is given in [17]. Again, data are provided by the Synapse organization in a structured schema where each table corresponds to an entity or a set of variables. JSON is the selected format for the data timeseries and the respective metadata and m4a extension is used to store the collected voice recordings. All the data will be fully available after following the 5-step access protocol of the Synapse organization and some additional steps for accessing MDS-UPDRS and PDQ-8 survey data. All the available data derive from the first six months of the ongoing data collection process of the study. It is worth mentioning that this dataset is suitable for diagnosis, as it contains data from both PD patients and healthy controls, as well as for severity estimation by setting MDS-UPDRS scores and PDQ-8 responses as the target variables. Another problem that could be possibly addressed is the medication-related fluctuations detection, as medication logs are also available and PD patients were prompted to provide input while being both on and off medication during the day.

The mPower dataset has already been used by many research papers. Firstly, some researchers detect PD based on the walking and standing data. For this cause, Zhang et al. [38] train a CNN, while Abujrida et al. [87] obtain 95% accuracy with the help of random forests. In the last approach, PD severity in terms of MDS-UPDRS scores is also estimated. Moreover, there are several studies that make use of the voice recordings to diagnose PD. For example, Zhang et al. [88] achieve 94.5% accuracy by deploying a CNN while Tougui et al. [52] train an extreme gradient boosting (XGBoost) model, obtaining 95.78% accuracy.

Furthermore, the two previous datasets have been combined in the context of PD Digital Biomaker DREAM challenge [89]. Each dataset has been leveraged to address a separate subtask. More specifically, the inertial measurements from the walking activity of the mPower study, as described above, were used to diagnose PD while the accelerometer data from the two smartwatches of the Levodopa Response Study Part I were used to estimate PD severity with respect to action tremor, as well as dyskinesia and bradykinesia presence. Regarding PD diagnosis, deep neural networks (DNNs) outperformed other approaches while regarding severity estimation, random forests proved one of the best performing models [40].

The Clinician Input Study in PD (CIS-PD) is an ongoing longitudinal study supported by the Michael J. Fox Foundation [90]. 51 PD patients have been enrolled and are requested to use the FOX wearable companion application and wear an Apple Watch during some hours of the day. Medication intake and symptoms severity are self-reported by the patients with the help of the application. Daily activity and nighttime movement measurements are also collected. The patients' monitoring lasts 6 months and during this period of time several periodical appointments to the doctor are scheduled to assess the observational results obtained from the application [89]. The feasibility and utility of this intervention is further discussed in [91]. Having PD severity labels for several timestamps during the 6-month period at our disposal supports the prediction of the deterioration or improvement of the PD symptoms, which is in accordance with what we are planning to do in the PD pilot of the ALAMEDA project.

Lonini et al. [43] leverage a part of this dataset and attempt to detect tremor and bradykinesia with CNNs and other statistical ensemble methods. In this study, 20 PD patients participate and are requested to wear 6 flexible skin-mounted MC10 BioStampRC sensors, while performing 13 common tasks, like walking or typing. In the same vein, Shawen et al. [45] combine acceleration data from an Apple Watch Series 2 and one skin-mounted sensor, identical with the previous, placed at the hand dorsal. Motor data are collected from 13 PD patients while performing 13 standardized motor tasks in certain repetitions. Demographic and clinical data, including MDS-UPDRS part III scores (0-4 scale) when the subjects are both on and off medication for two days are also provided. In this study, besides detecting tremor and bradykinesia, their severity level is estimated with the help of random forests. As stated by the authors in the previous three publications, this dataset is not publicly available yet, but we may access the data upon reasonable request to the authors and permission from the MJFF.

“EMA and wearable sensor monitoring in PD” is a publicly available dataset via DataverseNL which contains both objective sensory data and subjective EMA data. MOX5 sensors are deployed to collect accelerometer and gyroscope data from both wrists and the chest from 20 PD patients during 14 consecutive days. At the same time, EMA data are collected with the PsyMate smartphone application. Demographic and clinical data, including Hoehn and Yahr (H&Y) scores, levodopa equivalent daily dosage in mg and the presence or absence of motor fluctuations are also available. Sensory data are provided in European Data Format (EDF) files in a folder structure per patient while EMA data are provided in one CSV file for all the patients. The size of the total dataset reaches up to 99.9 GB. This dataset is suitable for H&Y scores estimation, self-reported motor state estimation, considering EMA data as ground truth data and motor fluctuations presence/absence detection. In [47], this dataset is presented more thoroughly and the authors further train a logistic regression model to detect medication-state fluctuations.

From the other hand, PD-BIOSTAMPRC21 is a PD dataset which is provided by the Institute of Electrical and Electronics Engineers (IEEE) DataPort, requiring only an account to access its data. This dataset contains acceleration signals related to gait, tremor and other movements or motor symptoms collected with MC10 BioStampRC sensors placed at the trunk, both thighs and both forearms from 17 PD patients and 17 controls (gait tremor and other movements). Each acceleration recording is accompanied by an annotation in terms of the type of the performed MDS-UPDRS assessment and its duration, as well as the patient’s medication status (on/off) when performing it. For each patient and each distinct sensor/placement the respective accelerometer data are stored in a CSV file. The size of the sensory data is 56GB approximately. Demographic and clinical data, including rest tremor Unified PD Rating Scale (UPDRS) scores are also provided in a single CSV file for all the participants. In conclusion, this dataset is suitable for PD diagnosis, severity estimation, in terms of UPDRS scores estimation and medication status (on/off) classification. We should note that these data were collected in the context of a research study, but the respective paper has not been published yet. Moreover, no other research articles that make use of this dataset were identified. However, there are many studies in the literature that propose ML models, trained on other data, to address the aforementioned problems.

The Daphnet Freezing of Gait (FoG) Dataset [92] is a completely open dataset provided via the University of California, Irvine (UCI) ML Repository. Signals from 3 triaxial accelerometer sensors placed on the trunk, the shank and the thigh of 10 PD patients are collected in the laboratory while the patients are performing walking tasks and activities of daily living (ADLs). These raw sensory data are available in txt files per patient (total files size: 85 MB approximately). Demographic and clinical data, including H&Y scale scores and medication status (on/off) during the experiments, are also available in a table in the respective publication [93], where the dataset is introduced. This dataset is clearly suitable for FoG detection and prediction, as there are 3 annotation classes (0: movement which is not part of the experiment, 1: movement which is part of the experiment and does not correspond to a FoG episode, like standing, walking and turning, 3: FoG episode).

The authors in [93] also proposed a system that detects FoG events online with 73.1% average sensitivity. The Daphnet dataset is very popular for FoG detection and prediction and besides this initial approach, many researchers have proposed various ML and DL models to address these problems. For example, Torvi et al. [94] make use of LSTMs and RNNs, obtaining 85%-95% accuracy while Li et al. [95] combine a CNN with an attention-enhanced LSTM improving significantly the classification performance. Very satisfactory results are also obtained by conventional ML approaches, like SVMs [96] with RBF kernels and the kNN algorithm [97]. Finally, Arami et al. [98] combine timeseries forecasting techniques, like autoregressive (AR) and autoregressive moving average (ARMA) with ML models to predict FoG. This is the only open dataset that we have out our disposal and enables FoG detection and prediction. However, even if we are not planning to identify FoG episodes, we could take advantage of these gait sequences for pretraining purposes, as well as for estimating PD patients' severity level, in terms of H&Y scale scores.

Gait in PD Dataset [99] is an open dataset shared by the PhysioNet organization [100]. It contains ground reaction force measurements from 8 insole sensors (Ultraflex Computer DynoGraphy, Infotronic Inc.) placed on each foot and the total force applied on each foot while walking. Demographic and clinical data, including H&Y scale scores, UPDRS scores, motor section of the UPDRS (UPDRSm) scores , Timed Up and Go (TUaG) tests results are also available in a txt file for all the subjects together. The total dataset has originated by merging data from three unique studies [101]–[103]. As a result, the final data are produced by 93 PD patients and 73 healthy controls. Force signals are organized in txt files per patient, leading to a total dataset size of 288.4 MB. In case we deploy insole pressure sensors in the PD pilot, this dataset will prove very useful. It is suitable for PD diagnosis, as both PD patients and healthy controls are enrolled, as well as for severity estimation, in terms of H&Y scale, UPDRS and UPDRSm scores and TUaG test results. Many studies in the literature make use of this dataset for gait dynamics analysis in PD. For example, DNNs, like feed-forward NNs, CNNs and LSTMS are trained to diagnose PD and estimate PD patients' H&Y scale scores [104]–[107]. For the same cause, simpler ML models, like SVMs, kNN, decision trees and random forests have been evaluated, as well [108], [109].

In conclusion, regarding the PD case, there are 3 datasets (mPower, PD-BIOSTAMPRC21 and Gait in PD) that are suitable for differentiating PD patients from healthy controls. The rest datasets contain biosignals solely from PD patients and can be used for the estimation of the current status of the disease and the severity of its manifestations, as well as for the prediction of the progression of the disease, in case there are ground truth data in a future timestamp, like e.g., in the Clinician Input Study in PD. Moreover, Daphnet is the only dataset that supports FoG episodes detection and prediction. All the included datasets are based on inertial sensors except for Gait in PD Dataset which is based on force sensors. Finally, only 3 datasets (EMA and wearable sensor monitoring in PD, Daphnet FoG Dataset and Gait in PD Dataset) are fully open and ready for use while all the others require some preliminary steps to access their data.

6.2.1.2 *Multiple Sclerosis*

In [110], a dataset of inertial signals for patient-reported fatigue assessment in MS is presented. 49 PwMS participate in this study, enabling data collection with Shimmer3 tri-axial accelerometers and gyroscopes mounted on the lateral side of the ankle of each foot, while performing a 6-minute walk test (6MWT). The Borg scale of perceived exertion is used for the ground truth data to quantify fatigue. Some statistical values of demographic and clinical information, including EDSS scores are also presented in a table in the discussed publication and the respective individual data for each patient are probably included in the original dataset. The dataset can be accessed upon request on the corresponding author of the paper. We can conclude that this dataset is suitable for self-reported fatigue quantification in the Borg scale and may also be used for MS degree of disability estimation, in terms of EDSS scores. The authors in [110], have extracted spatio-temporal gait parameters that after normalization and PCA feed a random forest regression model to estimate fatigue level. The respective mean absolute error (MAE) is 1.38 points at the Borg scale.

The "Real-World Falls in MS" dataset [111] contains inertial and tracking data from a tri-axial accelerometer and time of flight sensors, provided by the MotioWare system with the help of its waist-worn tag. The data are obtained from 25 PwMS, who are monitored over an eight-week period during free-living conditions. Data are provided in two directories, which contain npz files, one for the IMU data and one for the tracking data. Each npz file corresponds to one-day monitoring of one patient and two NumPy arrays are saved in it, one with the raw IMU data or the position and velocity states, depending on the two directories and one with the respective UTC timestamps. A csv file with the ground truth data (falls timestamps and patients id) is also available. This dataset is shared via the IEEE DataPort and registration to this platform is required to access them. The authors in [112] propose a context-aware fall detection system based on these data. Firstly, an auto-encoder detects fall candidates using reconstruction error of accelerometer signals followed by a hyper-ensemble of balanced random forests trained using both acceleration and movement features. The overall obtained sensitivity is 92.14%.

6.2.1.3 *Stroke*

6.2.1.4 *Healthy Population*

Additionally, we have identified some other datasets which consist of various biosignals collected from healthy people, that can be leveraged for sleep features extraction. All these datasets are based on smartwatches data and especially some of them make use of the Fitbit device that we are going to deploy in the ALAMEDA project, as well.

Firstly, in [113] the authors classify sleep stages (deep sleep, light sleep, REM sleep, wakefulness) based on Fitbit Charge 2 signals collected form 23 healthy subjects during 3 nights. The respective dataset is publicly available on a GitHub repository [114]. It also contains demographic data and self-reported indices of sleep quality, like the Pittsburgh Sleep Quality Index (PSQI) scores. There is one csv file for each subject and the total size of the dataset is 900 KB, approximately. The combination of a SVM with a XGBoost model in a two-level classification approach outperforms the rest approaches.

"Motion and heart rate from a wrist-worn wearable and labeled sleep from polysomnography" [115] is another publicly available dataset through PhysioNet [100]. Acceleration and heart rate signals are collected with an Apple watch embedded triaxial micro electro-mechanical system (MEMS) accelerometer and photoplethysmography (PPG), respectively, from 31 subjects. All the subjects wore the Apple Watch for 7-14 days to enable their ambulatory steps collection and on the last day, they spent the night in the lab for an eight-hour sleep opportunity. Acceleration and heart rate signals were collected while they were sleeping while at the same time ground truth data and sleep stages labels were acquired from the polysomnography. For each subject, there are 4 txt files, one for the acceleration data, one for the heart rate signals, one for the steps count and one for the labelled sleep stages (wake, N1, N2, N3, REM). The total size of the dataset is 2.2 GB. Furthermore, in [27], the authors deploy NNs to categorize sleep stages. For wake/sleep classification 90% accuracy is obtained, while for wake/NREM/REM classification 72% accuracy is obtained.

Moreover, some raw Fitbit data are provided from 30 healthy people via Zenodo [116]. These tracker data include minute-level output for physical activity, heart rate and sleep monitoring data from a 2-days period. There is one zip file for each day, which contains 11 csv files, one for each type of measurements. This dataset is publicly available and its total size is 600 MB, approximately. Additionally, in the context of the Digital Signals in Chronic Pain (DiSCover) Project, a 1-year longitudinal case-control observational study, self-reported health outcomes and quality of life data, behavioral data, lifestyle, genetic and environmental data are collected. More specifically, socio-demographic information are provided once during the enrollment phase, lifestyle and medication changes every month, Patient Health Questionnaire-9 (PHQ-9) scores every 3 months and step and sleep data are collected with the Fitbit wearable tracker throughout the study from 10,036 individuals. All the data are publicly available via Zenodo under a Creative Commons License [117].

6.2.2 Datasets Based on Images and Videos for FEA

There are also various datasets which contain images and videos of healthy people and support emotion recognition.

The CMU Multi-PIE Database [118] contains more than 750,000 face images from 337 subjects captured by 15 cameras in different view points and 19 different illumination conditions with the help of 18 flashes. The database also contains high resolution frontal images captured by a Canon EOS 10D camera. These data were acquired during 4 sessions in a 5-month period. In each session, subjects were required to display 7 different facial expressions (neutral, smile, anger, disgust, squint, scream, surprise). The total size of this dataset is 305 GB and its images are available through Flintbox after a relative request.

The MMI-Facial Expression database [119] contains 2900 videos and high-resolution images from 75 subjects, while they are displaying 7 different expressions (anger, disgust, fear, happiness, sadness, surprise, neutral). These recordings capture the full temporal pattern of a facial expression (neutral, onset, apex, offset phases and back to neutral). There are more than 300 respective manual annotations. Additionally, this database also contains expressions with a single Facial Action Coding System (FACS) Action Unit (AU) activated. More information about this database is given in [120], [121]. To retrieve these data, we need to request an account and log in.

FER-2013 [122] is an open dataset shared in Kaggle. It consists of 35,887 greyscale images retrieved from Google image search, where 7 different expressions (anger, disgust, fear, happiness, sadness, surprise and normal) are depicted.

The Japanese Female Facial Expression (JAFFE) [123] consists of 213 greyscale images with 7 different posed facial expressions (anger, disgust, fear, happiness, sadness, surprise and normal) from 10 subjects.

Each image has averaged semantic ratings on 6 facial expressions by 60 Japanese viewers. To access this dataset, certain instructions must be followed and a request must be submitted.

AffectNet [124] contains more than 1M images retrieved from the Web. 440,000 of them approximately are manually annotated in terms of 11 expressions/categories (neutral, happy, sad, surprise, fear, disgust, anger, contempt, none, uncertain, non-face). All the data are available upon request and their total size is 120 GB. More information about this dataset can be found in [125]. We should mention that various DL approaches based on this dataset have been proposed for expression analysis and emotion recognition.

The Radboud Faces Database (RaFD) [126] consists of 30,000 real-world images from 67 models trained with FACS to display 8 expressions (anger, disgust, fear, happiness, sadness, surprise, contempt and normal). Each expression is shown in three different gaze directions and is captured from 5 different camera angles. Similarly, the Real-World Affective Faces Database (RAF-DB) [127] contains 29,672 images annotated via crowdsourcing, in terms of 7 basic expressions and 12 further compound expressions. In both cases, access is provided for scientific research purposes upon request.

The Karolinska Directed Emotional Faces (KDEF) [128] is a dataset consisted of 4900 images from 70 subjects. 6 basic emotions are depicted, including anger, disgust, fear, happiness, sadness and surprise. The data are available to researchers upon request and more information about them can be found in [129].

The Binghamton University 3D Facial Expression (BU-3DFE) Database [130] consists of 2500 3D facial images, as captured by two points of view from 100 subjects. Each subject displays 7 different expressions (anger, disgust, fear, happiness, sadness, surprise and neutral) and all of them except for "neutral" have four levels of intensity, leading to 25 3D expressions models. The data are available upon request, after signing a usage agreement. The authors in [131] also propose a LDA model that performs this multi-class classification with 83% accuracy.

Additionally, BP4D-Spontaneous: Binghamton-Pittsburgh 3D Dynamic Spontaneous Facial Expression Database [130] contains both 2D and 3D videos of spontaneous/unposed facial expressions from 41 subjects. Each participant displays 8 different expressions (anger, disgust, fear, happiness, sadness, surprise, embarrass and physical pain). Manually annotated FACS AUs, automatically tracked head pose, and 2D/3D facial landmarks are also included. More information about this database can be found in [132], [133]. The total size of the dataset is 2.65 TB and the data are available upon request, after signing a usage agreement.

The previous dataset has also been combined with other biosignals, including heart rate, blood pressure, skin conductance-electrodermal activity (EDA), and respiration rate. It has also been extended by synchronizing 2D, 3D videos and thermal data, leading to the final form of the Multimodal Spontaneous Emotion database (BP4D+) [130]. Again, meta-data, like facial features and FACS codes are included. This time, 140 people display 10 different emotions. More information about this database are provide in [134]. The final data exceeds 10 TB in size and are available to researchers upon request. Moreover, the EB+ Dataset further expands the BP4D+ dataset by adding 2D videos and FACS annotations of facial expressions from 60 subjects [135]. The same steps should be followed to access these data. Finally, BU-EEG is another multimodal facial action database of a simultaneous collection of EEG signals and facial action videos of both posed expressions, AUs and spontaneous emotions from 29 subjects [136]. The same rules with the previous cases to access the data apply.

CASME II consists of 247 micro-expressions sequences labelled with AUs. 35 subjects display 5 different expressions (happy, disgust, surprise, regression and others) in a laboratory environment with appropriate illumination. This database is described more thoroughly in [137]. The authors also train a SVM model to address the 5-class classification problem, obtaining up to 63.41% accuracy.

GEneva Multimodal Emotion Portrayals (GEMEP) Facial expression Recognition and Analysis (FERA) is a subset of the GEMEP corpus [138], which was used in the FERA2011, the first FERA challenge. This subset contains 289 audiovisual emotion portrayals from 10 actors displaying 5 different expressions (anger, fear, sadness, relief, joy) [139]. These data are available for research purposes upon request.

Static Facial Expressions in the Wild contains 600 images from 68 subjects. This is a subset of static facial expression images extracted from the temporal facial expression database Dynamic Facial Expressions in the Wild, which consists of short facial video sequences extracted from movies. This dataset contains multiple faces in one scene, occlusions, different illumination conditions and ages, as well as different head poses (frontal and non-frontal) resembling real world scenarios. 7 different expressions (anger, disgust, fear, happiness, sadness, surprise and neutral) are displayed in the selected recordings. Access to this dataset may be gained upon request.

The Extended Cohn-Kanade (CK+) dataset consists of 593 videos of 7 posed expressions (anger, contempt, disgust, fear, happiness, sadness and surprise) from 123 subjects [140]. The respective data are available upon request.

The Oulu-CASIA near infrared (NIR) & visible light (VIS) facial expression database [141] contains 2880 videos with 6 different expressions (anger, disgust, fear, happiness, sadness and surprise) from 80 subjects, captured by the NIR and VIS imaging systems, in 3 different illumination conditions. These data are available after communication with the creator.

6.2.3 Datasets Based on Speech Recordings for Emotion Recognition

We have also identified some datasets that contain speech recordings, sometimes accompanied by videos or text, and can support emotion recognition tasks. All the included datasets are based on healthy populations and most of them correspond to the English language. However, there are a handful datasets on other languages, as well, including Italian and Greek, which are two languages under consideration for the development of the chatbot in the ALAMEDA project, as PwMS and PD patients are native speaker of Italian and Greek, respectively.

Firstly, we are going to present some datasets in the Italian language which corresponds to the MS pilot in the ALAMEDA project. For example, EMOVO is a publicly available dataset which contains audio recordings in Italian from 6 actors who played 14 sentences. Each one of these sentences is supposed to evoke one of the following 6 emotions: disgust, fear, anger, joy, surprise and sadness. The dataset is provided as a zip file through Google Drive and is described more thoroughly in [142]. In this study, the obtained accuracy for overall emotion recognition is 80%.

Moreover, the Database of Elicited Mood in Speech [143] is another audio dataset in Italian. It contains 9,365 emotional and 332 neutral samples from 68 native speakers. The included emotions are anger, sad, happy, fear, surprise, disgust and guilt. The dataset is available via the Zenodo organization upon request and more information about it can be found in the respective publication [144]. Similarly, the EmoFilm [145] is a multilingual speech corpus with 1115 audio clips from 43 films. The original films are in English, but the overdubbed Italian versions are included, as well. The selected clips correspond to 5 different emotions: anger, contempt, happiness, fear, and sadness. Again, the data are available via Zenodo upon request and more information about them can be found in the respective publication [146].

We have also identified one dataset in the Greek language which corresponds to the PD pilot in the ALAMEDA project. The Acted Emotional Speech Dynamic Database is a publicly available dataset via MEGA. It contains audio recordings from 5 actors with approximately 500 utterances, that correspond to 5 basic emotions (anger, disgust, fear, happiness and sadness). This dataset was initially presented in [147] and was further evaluated in [148]. The obtained accuracy in recognizing the intended emotion is approximately 74%.

One publicly available multimodal dataset, containing both audio and visual data, has been shared during the eNTERFACE of 2005, in the context of Project 2. 42 subjects participated in the 2-weeks experiments, that were conducted in English. They were asked to react to some short audio stories with the emotion that they considered most appropriate, leading to a final set of 1166 audio-video sequences, in which 6 different emotions (sad, angry, happy, fear, surprise, disgust) are depicted. More information about this dataset can be extracted from [53]. Similarly, the GEMEP Corpus [138] that we discussed in the previous section falls in this category, as well.

Another multimodal dataset, that contains both video and audio (in English) recordings is the Crowdsourced Emotional Multimodal Actors Dataset (CREMA-D), which is publicly available via a GitHub repository [55]. It contains 7,442 clips from 91 actors, who spoke some sentences, evoking each time the relative emotion, which is one of the 6 basic emotions anger, disgust, fear, happy, neutral and sad in 4 different intensity levels (low, medium, high, unspecified). The total size of this dataset is 7.55 GB approximately. A more thorough description of this dataset along with an initial evaluation of it can be found in [54].

MSP-IMPROV also contains audio and video recordings, which capture dyadic improvisations from 12 actors. A set of 20 sentences have been created that correspond to scenarios which evoke one of the 4 following emotions: angry, sad, happy and neutral. A detailed presentation of this corpus along with the evaluation of speech and facial emotion classifiers is provided in [149]. The dataset is available upon request.

Moreover, CMU-MOSI and CMU-MOSEI are two datasets which again contain both audio and video recordings and are publicly available via a GitHub repository [56]. Both are based in the English language. The first one contains 2199 opinion utterances with annotated sentiment (from very negative to very positive in 7 Likert steps) while the second contains audio-videos from more than 1000 speakers, that are annotated in terms of 6 basic emotions (happy, sad, anger, fear, disgust and surprise). In [57], the authors propose multi-attention RNNs to address the respective classification tasks.

Additionally, the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [150] is a publicly available dataset provided by the Zenodo organization. It contains 7,356 audio-video recordings in the English language from 24 actors which correspond to 7 different emotions (calm, happy, sad, angry, surprise, fear and disgust). More information about the dataset and its evaluation can be found in [151].

Finally, we have identified some speech datasets based on other languages. For example, the RECOLA [152] and the CaFE [153] datasets in French, the EMO-DB [154] in German, the Estonian Emotional Speech Corpus [155] obviously in Estonian, the ANAD in Arabic, the ShEMO dataset [156] in Persian, the URDU-Dataset [157] obviously in Urdu and the emotiontts open database [158] in Korean.

6.2.4 Datasets Based on Other Sources

PROMOPRO-MS is a dataset of patient-centered outcomes (PCOs) acquired from a cohort of people with MS progressively enrolled within an ongoing project funded by FISM. Up to date, more than 350 relapsing-remitting MS (RRMS) patients have been evaluated every four months through different PCOs covering physical, cognitive and psychosocial domains (i.e., manual ability, bladder functions, motor, cognitive, psychosocial fatigue, anxiety and depression, quality of life). For example, PCOs are provided in the following scales: EDSS, Functional Independence Measure (FIM), Hospital Anxiety and Depression Scale (HADS), Life Satisfaction Index, Modified Fatigue Impact Scale (MFIS), Montreal Cognitive Assessment (MoCA), Overactive Bladder Questionnaire (OAB-Q), Paced Auditory Serial Addition Test (PASAT 3) and Symbol Digit Modalities Test (SDMT) results. Moreover, general demographic and clinical information is collected (e.g., age, sex, weight, height, date of diagnosis, occurrence of relapse, rehabilitation treatments etc). We can access this dataset upon request to the medical partners of FISM, who are members of the ALAMEDA consortium.

The UK MS Register [159] combines data from clinical visits with data gathered as patient-reported outcomes (PROs), through a web portal. Access is granted upon request. Similarly, the Australian MS Longitudinal Study is a survey-based research study that has been running since 2001 and now has over 3000 people completing research surveys each year. The study is designed to provide data of practical use for improving the lives of Australians living with MS. The study collects and analyses self-reported data on matters of importance to people living with MS, including employment and the economic impact of MS, quality of life, service needs, energy use (for keeping cool), symptom severity and medication use. This data is used to better understand the challenges of living with MS and facilitate the provision of services and advocacy for people with MS.

References

[1] N. B. Ruparelia, 'Software development lifecycle models', ACM SIGSOFT Softw. Eng. Notes, vol. 35, no. 3, pp. 8–13, May 2010, doi: 10.1145/1764810.1764814.

[2] A. Mishra and D. Dubey, 'A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios', vol. 1, no. 5, p. 7, 2013.

[3] T. Bhuvaneswari and S. Prabaharan, 'A Survey on Software Development Life Cycle Models', Int. J. Comput. Sci. Mob. Comput..

[4] R. Kneuper, 'Sixty Years of Software Development Life Cycle Models', IEEE Ann. Hist. Comput., vol. 39, no. 3, pp. 41–54, 2017, doi: 10.1109/MAHC.2017.3481346.

[5] S. M. Salve, S. N. Samreen, and N. Khatri-Valmik, 'A Comparative Study on Software Development Life Cycle Models', vol. 05, no. 02, p. 6.

[6] W. W. Royce, 'Managing the development of large software systems: concepts and techniques', in Proceedings of the 9th international conference on Software Engineering, 1987, pp. 328–338.

[7] K. Petersen, C. Wohlin, and D. Baca, 'The Waterfall Model in Large-Scale Development', in Product-Focused Software Process Improvement, Berlin, Heidelberg, 2009, pp. 386–400. doi: 10.1007/978-3-642-02152-7_29.

[8] C. Larman and V. R. Basili, 'Iterative and incremental developments. a brief history', Computer, vol. 36, no. 6, pp. 47–56, Jun. 2003, doi: 10.1109/MC.2003.1204375.

[9] B. Boehm, 'A spiral model of software development and enhancement', ACM SIGSOFT Softw. Eng. Notes, vol. 11, no. 4, pp. 14–24, 1986.

[10] B. W. Boehm, 'A spiral model of software development and enhancement', Computer, vol. 21, no. 5, pp. 61–72, May 1988, doi: 10.1109/2.59.

[11] B. Boehm and W. J. Hansen, 'Spiral Development: Experience, Principles, and Refinements', Defense Technical Information Center, Fort Belvoir, VA, Jul. 2000. doi: 10.21236/ADA382590.

[12] M. Fowler and J. Highsmith, 'The Agile Manifesto', p. 7.

[13] 'Manifesto for Agile Software Development', p. 10.

[14] J. Highsmith and A. Cockburn, 'Agile software development: the business of innovation', Computer, vol. 34, no. 9, pp. 120–127, Sep. 2001, doi: 10.1109/2.947100.

[15] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, 'Agile Software Development Methods: Review and Analysis', ArXiv170908439 Cs, Sep. 2017, Accessed: Dec. 17, 2021. [Online]. Available: http://arxiv.org/abs/1709.08439

[16] ALAMEDA document “WP3-Activity1.2-Alert-display-options” (https://docs.google.com/spreadsheets/d/1TsAXC8nyqK_vpEdsobtBaml8Qq3j4dyVLhFF60zqHrw/edit#gid=0)

[17] ALAMEDA document “WP3 - Activity 1.4 - Intra-patient progress/regress prediction targets” (https://docs.google.com/document/d/1nIsWv7NqbePuBcsr5dj7soPNaXd9TxE7-OXR70ttqXc/edit#heading=h.hfrslyrtctsy)

[18] WP3-Activity1.2-Alert-display-options (https://docs.google.com/spreadsheets/d/1TsAXC8nyqK_vpEdsobtBaml8Qq3j4dyVLhFF60zqHrw/edit#gid=1704266690)

[19] Perceiver IO: A General Architecture for Structured Inputs & Outputs (https://paperswithcode.com/paper/perceiver-io-a-general-architecture-for)

[20] CRAN – Package tsfeatures (https://cran.r-project.org/web/packages/tsfeatures/index.html)

[21] Kats, a kit to analyze time series (https://github.com/facebookresearch/Kats)

[22] Sktime - A unified framework for machine learning with time series (https://www.sktime.org/en/stable/)

[23] Pyts - A Python package for time series classification (https://pythonrepo.com/repo/johannfaouzi-pyts-python-machine-learning)

[24] TODS: Automated Time-series Outlier Detection System (https://github.com/datamllab/tods)

[25] DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series (chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fwww.dfki.de%2Ffileadmin%2Fuser_upload%2Fimport%2F10175_DeepAnt.pdf&clen=2209388&chunk=true)

[26] Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning (https://arxiv.org/abs/1905.13628)

[27] O. Walch, Y. Huang, D. Forger, and C. Goldstein, ‘Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device’, Sleep, vol. 42, no. 12, p. zsz180, Dec. 2019, doi: 10.1093/sleep/zsz180.

[28] O. Walch, ojwalch/sleep_accel. 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/ojwalch/sleep_accel

[29] O. Walch, sleep_classifiers. 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/ojwalch/sleep_classifiers

[30] O. Walch, actigraphy-scripts. 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/ojwalch/actigraphy-scripts

[31] B. M. Bot et al., ‘The mPower study, Parkinson disease mobile data collected using ResearchKit’, Sci. Data, vol. 3, no. 1, p. 160011, Mar. 2016, doi: 10.1038/sdata.2016.11.

[32] mPower App. Sage Bionetworks, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/Sage-Bionetworks/mPower

[33] ResearchKit Framework. ResearchKit, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/ResearchKit/ResearchKit

[34] AppCore. ResearchKit, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/ResearchKit/AppCore

[35] BridgeSDK for iOS. Sage Bionetworks, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/Sage-Bionetworks/Bridge-iOS-SDK

[36] Sage-Bionetworks/mPower-sdata. Sage Bionetworks, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/Sage-Bionetworks/mPower-sdata

[37] 'MJFF Levodopa Response Study - syn20681023 - Wiki'. https://www.synapse.org/#!Synapse:syn20681023/wiki/594679 (accessed Nov. 05, 2021).

[38] H. Zhang, K. Deng, H. Li, R. L. Albin, and Y. Guan, 'Deep Learning Identifies Digital Biomarkers for Self-Reported Parkinson's Disease', Patterns N. Y. N, vol. 1, no. 3, p. 100042, Jun. 2020, doi: 10.1016/j.patter.2020.100042.

[39] PDDB: Parkinson's Disease Digital Biomarker. Guan Lab, 2021. Accessed: Nov. 05, 2021. [Online]. Available: https://github.com/GuanLab/PDDB

[40] S. K. Sieberts et al., 'Crowdsourcing digital health measures to predict Parkinson's disease severity: the Parkinson's Disease Digital Biomarker DREAM Challenge', Npj Digit. Med., vol. 4, no. 1, pp. 1–12, Mar. 2021, doi: 10.1038/s41746-021-00414-7.

[41] L. Lonini, DREAM_PD. 2020. Accessed: Nov. 08, 2021. [Online]. Available: https://github.com/Luke3D/DREAM_PD

[42] M. Daeschler, J. Elm, E. Klintworth, M. Afek, S. Lazar, and T. Simuni, 'Clinician-Input Study (CIS-PD): how the Fox Wearable Companion Application can influence treatment and care in Parkinson's disease (P3.048)', Neurology, vol. 90, no. 15 Supplement, Apr. 2018, Accessed: Oct. 29, 2021. [Online]. Available: https://n.neurology.org/content/90/15_Supplement/P3.048

[43] L. Lonini et al., 'Wearable sensors for Parkinson's disease: which data are worth collecting for training symptom detection models', Npj Digit. Med., vol. 1, Dec. 2018, doi: 10.1038/s41746-018-0071-z.

[44] L. Lonini, Wearable sensors for Parkinson's disease: which data are worth collecting for training symptom detection models. 2020. Accessed: Nov. 08, 2021. [Online]. Available: https://github.com/Luke3D/CIS_PD-NDM

[45] N. Shawen et al., 'Role of data measurement characteristics in the accurate detection of Parkinson's disease symptoms using wearable sensors', J. NeuroEngineering Rehabil., vol. 17, no. 1, p. 52, Apr. 2020, doi: 10.1186/s12984-020-00684-4.

[46] N. Shawen, DataCharacteristics_PD. 2021. Accessed: Nov. 08, 2021. [Online]. Available: https://github.com/nshawen/DataCharacteristics_PD

[47] J. G. V. Habets et al., 'A Long-Term, Real-Life Parkinson Monitoring Database Combining Unscripted Objective and Subjective Recordings', Data, vol. 6, no. 2, Art. no. 2, Feb. 2021, doi: 10.3390/data6020022.

[48] J. Habets, sensor_EMA_PD_monitoring. 2021. Accessed: Nov. 08, 2021. [Online]. Available: https://github.com/jgvhabets/sensor_EMA_PD_monitoring

[49] T. Giannakopoulos, pyAudioAnalysis: Python Audio Analysis Library: Feature Extraction, Classification, Segmentation and Applications. Accessed: Nov. 07, 2021. [Online]. Available: https://github.com/tyiannak/pyAudioAnalysis

[50] T. Giannakopoulos, A Python library for audio feature extraction, classification, segmentation and applications. 2021. Accessed: Nov. 07, 2021. [Online]. Available: https://github.com/tyiannak/pyAudioAnalysis

[51] T. Giannakopoulos, 'pyaudioanalysis: An open-source python library for audio signal analysis', PloS One, vol. 10, no. 12, p. e0144610, 2015.

[52] I. Tougui, A. Jilbab, and J. E. Mhamdi, 'Analysis of Smartphone Recordings in Time, Frequency, and Cepstral Domains to Classify Parkinson's Disease', Healthc. Inform. Res., vol. 26, no. 4, pp. 274–283, Oct. 2020, doi: 10.4258/hir.2020.26.4.274.

[53] O. Martin, I. Kotsia, B. Macq, and I. Pitas, 'The eNTERFACE' 05 Audio-Visual Emotion Database', in 22nd International Conference on Data Engineering Workshops (ICDEW'06), Apr. 2006, pp. 8–8. doi: 10.1109/ICDEW.2006.145.

[54] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, 'CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset', IEEE Trans. Affect. Comput., vol. 5, no. 4, pp. 377–390, 2014, doi: 10.1109/TAFFC.2014.2336244.

[55] CheyneyComputerScience/CREMA-D. Cheyney Computer Science, 2021. Accessed: Nov. 01, 2021. [Online]. Available: https://github.com/CheyneyComputerScience/CREMA-D

[56] A. Zadeh, CMU-Multimodal SDK Version 1.2.0 (mmsdk). 2021. Accessed: Nov. 01, 2021. [Online]. Available: https://github.com/A2Zadeh/CMU-MultimodalSDK

[57] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency, 'Multi-attention Recurrent Network for Human Communication Comprehension', presented at the Thirty-Second AAAI Conference on Artificial Intelligence, Apr. 2018. Accessed: Nov. 01, 2021. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17390

[58] A. P. Creagh et al., 'Smartphone- and Smartwatch-Based Remote Characterisation of Ambulation in Multiple Sclerosis During the Two-Minute Walk Test', IEEE J. Biomed. Health Inform., vol. 25, no. 3, pp. 838–849, Mar. 2021, doi: 10.1109/JBHI.2020.2998187.

[59] A. Creagh, MS-GAIT_feature_extraction. 2021. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/apcreagh/MS-GAIT_feature_extraction

[60] A. P. Creagh, F. Lipsmeier, M. Lindemann, and M. D. Vos, 'Interpretable deep learning for the remote characterisation of ambulation in multiple sclerosis using smartphones', Sci. Rep., vol. 11, no. 1, p. 14301, Jul. 2021, doi: 10.1038/s41598-021-92776-x.

[61] A. Creagh, Interpretable Deep Learning for the Remote Characterisation of Ambulation in Multiple Sclerosis using Smartphones. 2021. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/apcreagh/MS-GAIT_InterpretableDL

[62] A. P. Creagh et al., 'Smartphone-based remote assessment of upper extremity function for multiple sclerosis using the Draw a Shape Test', Physiol. Meas., vol. 41, no. 5, p. 054002, Jun. 2020, doi: 10.1088/1361-6579/ab8771.

[63] A. Creagh, Draw a Shape Feature Extraction. 2021. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/apcreagh/DaS_feature_extraction

[64] karnajitsen, Multiple Sclerosis. 2018. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/karnajitsen/Multiple_Sclerosis

[65] N. Steinemann et al., 'The Swiss Multiple Sclerosis Registry (SMSR): study protocol of a participatory, nationwide registry to promote epidemiological and patient-centered MS research', BMC Neurol., vol. 18, no. 1, p. 111, Aug. 2018, doi: 10.1186/s12883-018-1118-0.

[66] MSRegistry Backend. uzh, 2019. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/uzh/msregistry

[67] A. Dobrasinovic, EDSS Calculator. 2021. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/adobrasinovic/edss

[68] A. R. Kumar, Wearable Inertial Sensors for Exergames. 2020. Accessed: Nov. 10, 2021. [Online]. Available: https://github.com/rashwinr/WISE

[69] A. Raj Kumar, S. Bilaloglu, P. Raghavan, and V. Kapila, 'Grasp Rehabilitator: A Mechatronic Approach', presented at the 2019 Design of Medical Devices Conference, Jul. 2019. doi: 10.1115/DMD2019-3242.

[70] A. R. Kumar, grasp-rehabilitator. 2020. Accessed: Nov. 10, 2021. [Online]. Available: https://github.com/rashwinr/grasp-rehabilitator

[71] S. Alves, BackOnTrack. 2019. Accessed: Nov. 10, 2021. [Online]. Available: https://github.com/SamuelIGT/BackOnTrack

[72] C. Velvin, 'Exergame for rehabilitering av balansen til slagpasienter', 2019, Accessed: Nov. 10, 2021. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2609654

[73] C. Velvin, P-LagMotSlag. 2019. Accessed: Nov. 10, 2021. [Online]. Available: https://github.com/camilve/P-LagMotSlag

[74] Ó. Örn, olafurorng/kayak-game-unity. 2020. Accessed: Nov. 10, 2021. [Online]. Available: https://github.com/olafurorng/kayak-game-unity

[75] 'Unterstützende Nachsorge durch Schlaganfall-Lotsen - STROKE OWL - Über das Projekt'. https://stroke-owl.de/de/startseite (accessed Nov. 11, 2021).

[76] LotsenApp 2. OFFIS e.V., 2021. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/offis/lotsen-app

[77] alim momin, BrainFit-Android-App. 2019. Accessed: Nov. 11, 2021. [Online]. Available: https://github.com/alimmomin018/BrainFit-Android-App

[78] Z. Pirani, N. Tasbi, R. Fakir, and A. Momin, 'Assistive application for the people with mini-brain stroke', in 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), Feb. 2017, pp. 1–6. doi: 10.1109/ICECCT.2017.8117816.

[79] Z. Pirani, N. Tasbi, R. Fakir, and A. Momin, 'Android based Assistive Toolkit for the Mini Brain Stroke Patients', Int. J. Comput. Appl., vol. 165, no. 13, pp. 17–21, May 2017.

[80] Wearable Sensors Dataset (https://docs.google.com/spreadsheets/d/1RtB3GzS5--qw182U1xxqVY0OPB4LpAYA/edit#gid=1732665831)

[81] Facial Expression Analysis Datasets (https://docs.google.com/spreadsheets/d/123j3pb1QZVtPXq6aypdUQJq3fw1Vnpbf/edit#gid=1946599440)

[82] Speech Emotions Recognition Analysis Datasets (https://docs.google.com/spreadsheets/d/1_k7o-ZX3yjLiVRnPr9d4y1vQVguaDq3J/edit#gid=952118311)

[83] MJFF Levodopa Response Study (https://www.synapse.org/#!Synapse:syn20681023/wiki/594680)

[84] J.-F. Daneault et al., 'Accelerometer data collected with a minimum set of wearable sensors from subjects with Parkinson's disease', Sci. Data, vol. 8, no. 1, p. 48, Feb. 2021, doi: 10.1038/s41597-021-00830-0.

[85] G. Vergara-Diaz et al., 'Limb and trunk accelerometer data collected with wearable sensors from subjects with Parkinson's disease', Sci. Data, vol. 8, no. 1, p. 47, Feb. 2021, doi: 10.1038/s41597-021-00831-z.

[86] mPower Mobile Parkinson Disease Study (https://www.synapse.org/#!Synapse:syn4993293/wiki/)

[87] H. Abujrida, E. Agu, and K. Pahlavan, ‘Machine learning-based motor assessment of Parkinson’s disease using postural sway, gait and lifestyle features on crowdsourced smartphone data’, Biomed. Phys. Eng. Express, vol. 6, no. 3, p. 035005, Mar. 2020, doi: 10.1088/2057-1976/ab39a8.

[88] H. Zhang, A. Wang, D. Li, and W. Xu, ‘DeepVoice: A voiceprint-based mobile health framework for Parkinson’s disease identification’, in 2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI), Mar. 2018, pp. 214–217. doi: 10.1109/BHI.2018.8333407.

[89] Parkinon’s Disease Digital Biomaker DREAM Challenge (https://www.synapse.org/#!Synapse:syn8717496/wiki/422884)

[90] MJFF Clinician Input Study in Parkinson’s Disease publication (https://www.michaeljfox.org/publication/michael-j-fox-foundation-launches-clinician-input-study-parkinsons-disease-evaluate)

[91] J. J. Elm et al., ‘Feasibility and utility of a clinician dashboard from wearable and mobile application Parkinson’s disease data’, Npj Digit. Med., vol. 2, no. 1, pp. 1–6, Sep. 2019, doi: 10.1038/s41746-019-0169-y.

[92] Daphnet Freezing of Gait Data Set UCI (https://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait)

[93] M. Bachlin et al., ‘Wearable Assistant for Parkinson’s Disease Patients With the Freezing of Gait Symptom’, IEEE Trans. Inf. Technol. Biomed., vol. 14, no. 2, pp. 436–446, Mar. 2010, doi: 10.1109/TITB.2009.2036165.

[94] V. G. Torvi, A. Bhattacharya, and S. Chakraborty, ‘Deep Domain Adaptation to Predict Freezing of Gait in Patients with Parkinson’s Disease’, in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2018, pp. 1001–1006. doi: 10.1109/ICMLA.2018.00163.

[95] B. Li, Y. Sun, Z. Yao, J. Wang, S. Wang, and X. Yang, ‘Improved deep learning technique to detect freezing of gait in parkinson’s disease based on wearable sensors’, Electron. Switz., vol. 9, no. 11, pp. 1–20, 2020, doi: 10.3390/electronics9111919.

[96] N. Kleanthous, A. J. Hussain, W. Khan, and P. Liatsis, ‘A new machine learning based approach to predict Freezing of Gait’, Pattern Recognit. Lett., vol. 140, pp. 119–126, 2020, doi: 10.1016/j.patrec.2020.09.011.

[97] A. Halder, R. Singh, A. Suri, and D. Joshi, ‘Predicting State Transition in Freezing of Gait via Acceleration Measurements for Controlled Cueing in Parkinson’s Disease’, IEEE Trans. Instrum. Meas., vol. 70, pp. 1–16, 2021, doi: 10.1109/TIM.2021.3090153.

[98] A. Arami, A. Poulakakis-Daktylidis, Y. F. Tai, and E. Burdet, ‘Prediction of Gait Freezing in Parkinsonian Patients: A Binary Classification Augmented With Time Series Prediction’, IEEE Trans. Neural Syst. Rehabil. Eng. Publ. IEEE Eng. Med. Biol. Soc., vol. 27, no. 9, pp. 1909–1919, 2019, doi: 10.1109/TNSRE.2019.2933626.

[99] PhysioNet Gait in Parkinson’s Disease (https://physionet.org/content/gaitpdb/1.0.0/)

[100] A. L. Goldberger et al., ‘PhysioBank, PhysioToolkit, and PhysioNet’, Circulation, vol. 101, no. 23, pp. e215–e220, Jun. 2000, doi: 10.1161/01.CIR.101.23.e215.

[101] G. Yogev, N. Giladi, C. Peretz, S. Springer, E. S. Simon, and J. M. Hausdorff, ‘Dual tasking, gait rhythmicity, and Parkinson’s disease: Which aspects of gait are attention demanding?’, Eur. J. Neurosci., vol. 22, no. 5, pp. 1248–1256, 2005, doi: 10.1111/j.1460-9568.2005.04298.x.

[102] J. M. Hausdorff, J. Lowenthal, T. Herman, L. Gruendlinger, C. Peretz, and N. Giladi, 'Rhythmic auditory stimulation modulates gait variability in Parkinson's disease', Eur. J. Neurosci., vol. 26, no. 8, pp. 2369–2375, 2007, doi: 10.1111/j.1460-9568.2007.05810.x.

[103] S. Frenkel-Toledo, N. Giladi, C. Peretz, T. Herman, L. Gruendlinger, and J. M. Hausdorff, 'Treadmill walking as an external pacemaker to improve gait rhythm and stability in Parkinson's disease', Mov. Disord., vol. 20, no. 9, pp. 1109–1114, 2005, doi: 10.1002/mds.20507.

[104] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, 'Early Detection of Parkinson Disease using Deep Neural Networks on Gait Dynamics', in 2020 International Joint Conference on Neural Networks (IJCNN), Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207380.

[105] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, 'Deep 1D-Convnet for accurate Parkinson disease detection and severity prediction from gait', Expert Syst. Appl., vol. 143, 2020, doi: 10.1016/j.eswa.2019.113075.

[106] Y. Xia, Z. Yao, Q. Ye, and N. Cheng, 'A Dual-Modal Attention-Enhanced Deep Learning Network for Quantification of Parkinson's Disease Characteristics', IEEE Trans. Neural Syst. Rehabil. Eng., vol. 28, no. 1, pp. 42–51, Jan. 2020, doi: 10.1109/TNSRE.2019.2946194.

[107] Y. Nancy Jane, H. Khanna Nehemiah, and K. Arputharaj, 'A Q-backpropagated time delay neural network for diagnosing severity of gait disturbances in Parkinson's disease', J. Biomed. Inform., vol. 60, pp. 169–176, 2016, doi: 10.1016/j.jbi.2016.01.014.

[108] B. E, B. D, V. K. Elumalai, and U. K, 'Data-driven gait analysis for diagnosis and severity rating of Parkinson's disease', Med. Eng. Phys., vol. 91, pp. 54–64, May 2021, doi: 10.1016/j.medengphy.2021.03.005.

[109] N. Khoury, F. Attal, Y. Amirat, L. Oukhellou, and S. Mohammed, 'Data-driven based approach to aid Parkinson's disease diagnosis', Sens. Switz., vol. 19, no. 2, 2019, doi: 10.3390/s19020242.

[110] A. A. Ibrahim, A. Küderle, H. Gaßner, J. Klucken, B. M. Eskofier, and F. Kluge, 'Inertial sensor-based gait parameters reflect patient-reported fatigue in multiple sclerosis', J. NeuroEngineering Rehabil., vol. 17, no. 1, p. 165, Dec. 2020, doi: 10.1186/s12984-020-00798-9.

[111] C. Mosquera-Lopez, 'Real-world falls in Multiple Sclerosis (MS)'. IEEE, Dec. 08, 2020. Accessed: Nov. 01, 2021. [Online]. Available: https://ieee-dataport.org/open-access/real-world-falls-multiple-sclerosis-ms

[112] C. Mosquera-Lopez et al., 'Automated Detection of Real-World Falls: Modeled from People with Multiple Sclerosis', IEEE J. Biomed. Health Inform., pp. 1–1, 2020, doi: 10.1109/JBHI.2020.3041035.

[113] Z. Liang and M. A. Chapa-Martell, 'A Multi-Level Classification Approach for Sleep Stage Prediction With Processed Data Derived From Consumer Wearable Activity Trackers', Front. Digit. Health, vol. 3, p. 49, 2021, doi: 10.3389/fdgth.2021.665946.

[114] PiranitaGomez, Fitbit_sleepStaging_multiLevelML. 2021. Accessed: Nov. 01, 2021. [Online]. Available: https://github.com/PiranitaGomez/Fitbit_sleepStaging_multiLevelML

[115] O. Walch, 'Motion and heart rate from a wrist-worn wearable and labeled sleep from polysomnography'. physionet.org. doi: 10.13026/HMHS-PY35.

[116] R. Furberg, J. Brinton, M. Keating, and A. Ortiz, 'Crowd-sourced Fitbit datasets 03.12.2016-05.12.2016'. Zenodo, May 31, 2016. doi: 10.5281/zenodo.53894.

[117] M. Makhmutova, R. Kainkaryam, M. Ferreira, J. Min, M. Jaggi, and I. Clay, 'PSYCHE-D: predicting change in depression severity using person-generated health data (DATASET)'. Zenodo, Jul. 09, 2021. doi: 10.5281/zenodo.5085146.

[118] 'The CMU Multi-PIE Face Database'. http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html (accessed Nov. 01, 2021).

[119] 'MMI Facial Expression Database - Home'. https://mmifacedb.eu/ (accessed Nov. 01, 2021).

[120] M. Valstar and M. Pantic, 'Induced disgust, happiness and surprise: an addition to the mmi facial expression database', in Proc. 3rd Intern. Workshop on EMOTION (satellite of LREC): Corpora for Research on Emotion and Affect, 2010, p. 65.

[121] M. Pantic, M. Valstar, R. Rademaker, and L. Maat, 'Web-based database for facial expression analysis', in 2005 IEEE International Conference on Multimedia and Expo, Jul. 2005, p. 5 pp.-. doi: 10.1109/ICME.2005.1521424.

[122] 'FER-2013'. https://kaggle.com/msambare/fer2013 (accessed Nov. 01, 2021).

[123] M. Lyons, M. Kamachi, and J. Gyoba, 'The Japanese Female Facial Expression (JAFFE) Dataset'. Zenodo, Apr. 14, 1998. doi: 10.5281/zenodo.3451524.

[124] 'AffectNet – Mohammad H. Mahoor, PhD'. http://mohammadmahoor.com/affectnet/ (accessed Nov. 01, 2021).

[125] A. Mollahosseini, B. Hasani, and M. H. Mahoor, 'AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild', IEEE Trans. Affect. Comput., vol. 10, no. 1, pp. 18–31, Jan. 2019, doi: 10.1109/TAFFC.2017.2740923.

[126] O. Langner, R. Dotsch, G. Bijlstra, D. H. J. Wigboldus, S. T. Hawk, and A. van Knippenberg, 'Presentation and validation of the Radboud Faces Database', Cogn. Emot., vol. 24, no. 8, pp. 1377–1388, Dec. 2010, doi: 10.1080/02699930903485076.

[127] S. Li and W. Deng, 'Reliable Crowdsourcing and Deep Locality-Preserving Learning for Unconstrained Facial Expression Recognition', IEEE Trans. Image Process., vol. 28, no. 1, pp. 356–370, Jan. 2019, doi: 10.1109/TIP.2018.2868382.

[128] 'KDEF & AKDEF'. https://www.kdef.se/ (accessed Nov. 01, 2021).

[129] D. Lundqvist, A. Flykt, and A. Öhman, 'The Karolinska directed emotional faces (KDEF)', CD ROM Dep. Clin. Neurosci. Psychol. Sect. Karolinska Institutet, vol. 91, no. 630, pp. 2–2, 1998.

[130] '3D Facial Expression Database - Binghamton University'. http://www.cs.binghamton.edu/~lijun/Research/3DFE/3DFE_Analysis.html (accessed Nov. 01, 2021).

[131] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, 'A 3D facial expression database for facial behavior research', in 7th International Conference on Automatic Face and Gesture Recognition (FGR06), Apr. 2006, pp. 211–216. doi: 10.1109/FGR.2006.6.

[132] X. Zhang et al., 'A high-resolution spontaneous 3D dynamic facial expression database', in 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), Apr. 2013, pp. 1–6. doi: 10.1109/FG.2013.6553788.

[133] X. Zhang et al., 'BP4D-Spontaneous: a high-resolution spontaneous 3D dynamic facial expression database', Image Vis. Comput., vol. 32, no. 10, pp. 692–706, Oct. 2014, doi: 10.1016/j.imavis.2014.06.002.

[134] Z. Zhang et al., ‘Multimodal Spontaneous Emotion Corpus for Human Behavior Analysis’, 2016, pp. 3438–3446. Accessed: Nov. 01, 2021. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Zhang_Multimodal_Spontaneous_Emotion_CVPR_2016_paper.html

[135] I. O. Ertugrul, J. F. Cohn, L. A. Jeni, Z. Zhang, L. Yin, and Q. Ji, ‘Cross-domain AU Detection: Domains, Learning Approaches, and Measures’, in 2019 14th IEEE International Conference on Automatic Face Gesture Recognition (FG 2019), May 2019, pp. 1–8. doi: 10.1109/FG.2019.8756543.

[136] X. Li, X. Zhang, H. Yang, W. Duan, W. Dai, and L. Yin, ‘An EEG-Based Multi-Modal Emotion Database with Both Posed and Authentic Facial Actions for Emotion Analysis’, in 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), Nov. 2020, pp. 336–343. doi: 10.1109/FG47880.2020.00050.

[137] W.-J. Yan et al., ‘CASME II: An Improved Spontaneous Micro-Expression Database and the Baseline Evaluation’, PLOS ONE, vol. 9, no. 1, p. e86041, 2014, doi: 10.1371/journal.pone.0086041.

[138] ‘GEMEP Corpus - Swiss Center For Affective Sciences - UNIGE’, Jun. 29, 2016. https://www.unige.ch/cisa/gemep (accessed Nov. 01, 2021).

[139] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer, ‘The first facial expression recognition and analysis challenge’, in Face and Gesture 2011, Santa Barbara, CA, USA, Mar. 2011, pp. 921–926. doi: 10.1109/FG.2011.5771374.

[140] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, ‘The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression’, in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, Jun. 2010, pp. 94–101. doi: 10.1109/CVPRW.2010.5543262.

[141] ‘Oulu-CASIA NIR&VIS facial expression database’. https://www.oulu.fi/cmvs/node/41316 (accessed Nov. 01, 2021).

[142] G. Costantini, I. Iaderola, A. Paoloni, and M. Todisco, ‘EMOVO corpus: an Italian emotional speech database’, in International Conference on Language Resources and Evaluation (LREC 2014), 2014, pp. 3501–3504.

[143] E. Parada-Cabaleiro, G. Costantini, A. Batliner, M. Schmitt, and B. Schuller, ‘DEMoS: an Italian emotional speech corpus. Elicitation methods, machine learning, and perception’. Feb. 22, 2019. doi: 10.1007/s10579-019-09450-y.

[144] E. Parada-Cabaleiro, G. Costantini, A. Batliner, M. Schmitt, and B. W. Schuller, ‘DEMoS: an Italian emotional speech corpus’, Lang. Resour. Eval., vol. 54, no. 2, pp. 341–383, Jun. 2020, doi: 10.1007/s10579-019-09450-y.

[145] E. Parada-Cabaleiro, G. Costantini, A. Batliner, A. Baird, and B. Schuller, ‘EmoFilm - A multilingual emotional speech corpus’. Zenodo, Sep. 06, 2018. doi: 10.5281/zenodo.1326428.

[146] E. Parada-Cabaleiro, G. Costantini, A. Batliner, A. Baird, and B. Schuller, ‘Categorical vs dimensional perception of italian emotional speech’, 2018.

[147] N. Vryzas, R. Kotsakis, A. Liatsou, C. A. Dimoulas, and G. Kalliris, ‘Speech Emotion Recognition for Performance Interaction’, J. Audio Eng. Soc., vol. 66, no. 6, pp. 457–467, Jun. 2018.

[148] N. Vryzas, M. Matsiola, R. Kotsakis, C. Dimoulas, and G. Kalliris, ‘Subjective Evaluation of a Speech Emotion Recognition Interaction Framework’, in Proceedings of the Audio Mostly 2018 on Sound

in Immersion and Emotion, New York, NY, USA, Sep. 2018, pp. 1–7. doi: 10.1145/3243274.3243294.

[149] C. Busso, S. Parthasarathy, A. Burmania, M. AbdelWahab, N. Sadoughi, and E. M. Provost, 'MSP-IMPROV: An Acted Corpus of Dyadic Interactions to Study Emotion Perception', IEEE Trans. Affect. Comput., vol. 8, no. 1, pp. 67–80, Jan. 2017, doi: 10.1109/TAFFC.2016.2515617.

[150] S. R. Livingstone and F. A. Russo, 'The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)'. Zenodo, Apr. 05, 2018. doi: 10.5281/zenodo.1188976.

[151] S. R. Livingstone and F. A. Russo, 'The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English', PLOS ONE, vol. 13, no. 5, p. e0196391, 2018, doi: 10.1371/journal.pone.0196391.

[152] F. Ringeval, A. Sonderegger, J. Sauer, and D. Lalanne, 'Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions', in 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), Apr. 2013, pp. 1–8. doi: 10.1109/FG.2013.6553805.

[153] P. Gournay, O. Lahaie, and R. Lefebvre, 'A canadian french emotional speech dataset', in Proceedings of the 9th ACM Multimedia Systems Conference, New York, NY, USA, Jun. 2018, pp. 399–402. doi: 10.1145/3204949.3208121.

[154] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, 'A database of German emotional speech.', in Interspeech, 2005, vol. 5, pp. 1517–1520.

[155] R. Altrov and H. Pajupuu, 'Estonian Emotional Speech Corpus: Release 1.', 2008, pp. 9–15. doi: 10.13140/RG.2.1.1110.4721.

[156] O. Mohamad Nezami, P. Jamshid Lou, and M. Karami, 'ShEMO: a large-scale validated database for Persian speech emotion detection', Lang. Resour. Eval., vol. 53, no. 1, pp. 1–16, Mar. 2019, doi: 10.1007/s10579-018-9427-x.

[157] S. Latif, A. Qayyum, M. Usman, and J. Qadir, 'Cross Lingual Speech Emotion Recognition: Urdu vs. Western Languages', in 2018 International Conference on Frontiers of Information Technology (FIT), Dec. 2018, pp. 88–93. doi: 10.1109/FIT.2018.00023.

[158] Y. Choi, Y. Jung, Y. Suh, and H. Kim, 'Perceptually Guided End-to-End Text-to-Speech With MOS Prediction', ArXiv201101174 Cs Eess, Aug. 2021, Accessed: Nov. 02, 2021. [Online]. Available: http://arxiv.org/abs/2011.01174

[159] 'Home | UK MS Register'. https://www.ukmsregister.org/ (accessed Nov. 01, 2021).