SEcure Decentralized Intelligent Data MARKetplace

D5.1 Evaluation methodology, metrics and integration plan

Document Identification	
Contractual delivery date:	30/09/2023
Actual delivery date:	23/11/2023
Responsible beneficiary:	WINGS
Contributing beneficiaries:	All
Dissemination level:	PU
Version:	1.0
Status:	Final

Keywords:

Development process, continuous integration, continuous delivery, datasets, use cases, software components integration, integration plan, supported scenarios, evaluation methodology, evaluation process, evaluation criteria, trial definitions, KPIs

This document is issued within the frame and for the purpose of the SEDIMARK project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070074. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

Document Identification			
Related WP	WP5	**Related Deliverables(s):**	-
Document reference:	SEDIMARK_D5.1	**Total number of pages:**	129129

List of Contributors	
Name	**Partner**
Panagiotis Vlacheas Grigorios Koutantos	WINGS
Luis Sánchez Jorge Lanza Pablo Sotres Juan Ramón Santana	UC
Elias Tragos Diarmuid O'Reilly Morgan Erika Duriakova	NUID-UCD
Tarek Elsaleh	SURREY
Alberto Carelli	LINKS
Arturo Medela Maxime Costa longa Jairo Rojas-Delgado	ATOS
Gabriel-Mihail Danciu	SIE
Maroua Bahri	INRIA
Nikos Babis Ioannis Tsogias	MYT
Luc Gasser Léa Robert Franck Le Gall Gilles Orazi	EGM

List of Contributors	
Name	**Partner**
Eero Jalo	FV
Juan Echeverria	SDR

Document History			
0.1	01/08/2023	WINGS	First draft of ToC
0.2	21/08/2023	NUID UCD	Contribution to Sections 4.1, 6.1, 6.2, 6.3, 6.4.5, 6.4.6
0.3	05/09/2023	WINGS	Merge contributions from all partners
0.4	19/09/2023	WINGS	Add additional contributions and refinements
0.5	02/10/2023	NUID UCD	Contributions to Section 5.2.5
0.6	10/10/2023	WINGS	First candidate version for internal review
0.7	26/10/2033	NUID UCD, LINKS	Addressed comments of the internal review
0.8	27/10/2023	WINGS	Final fixes based on internal review comments
0.9	17/11/2023	ATOS	Quality Review Form.
1.0	22/11/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 2	Alberto Carelli (LINKS)	21/10/2023
Reviewer 1	Elias Tragos, Erika Duriakova, Diarmuid O' Reilly Morgan (NUID UCD)	17/10/2023
Quality manager	María Guadalupe Rodríguez (ATOS)	21/11/2023
Project Coordinator	Arturo Medela (ATOS)	22/11/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / Acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BI	Baseline Infrastructure
BIFs	Baseline Infrastructure Facilitators
CI/CD	Continuous Integration and Continuous Delivery/Continuous Deployment
CKAN	Comprehensive Knowledge Archive Network
CLI	Command Line Interfaces
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DRACO	Data Reduction and Calibration Operation
DT	Digital Twin
Dx,y	Deliverable number y belonging to WP x
EC	European Commission
ECDSA	Elliptic Curve Digital Signature Algorithm
ERC	Environmental Regulation Commission
GIS	Geographic Information System
GUI	Graphical User Interface
GWs	Gateways
HRI	Helsinki Region Information
HSY	Helsinki Region Environmental Services
HTTPS	Hypertext Transfer Protocol Secure
IdM	Identity Management

Abbreviation / Acronym	Description
IEC	International Electrotechnical Commission
IM	Identity Management
IoT	Internet of Things
IOTA	Internet of Things Application
ISO	International Organization for Standardization
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
KPIs	Key Performance Indicators
LoRaWAN	Lo(ng) Ra(nge) Wide Area Network
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLOps	Machine Learning Operations
MSE	Mean Square Error
MVP	Minimum Viable Product
NAP	Network Access Point
NFT	Non-fungible tokens
NGSI-LD	Next Generation Service Interfaces for Linked Data
ODRL	Open Digital Rights Language
OS	Operating System
P2P	Peer-to-peer
PCA	Principal component Analysis
PDP	Policy Decision Point
PEP	Policy Enforcement Point
RDF	Resource Description Framework
PoC	Proof of Concept
RERUM	REliable, Resilient and secUre IoT for sMart city applications
RESTful	REpresentational State Transfer
SDI	Serial Digital Interface

Abbreviation / Acronym	Description
SHA	Secure Hash Algorithm
SPARQL	SPARQL Protocol and RDF Query Language
SQuaRE	Software Quality Assessments and Recommendations
SSCP	Santander Smart City Platform
SSI	Self-Sovereign Identity
SSL	Secure Sockets Layer
SVN	Software Virtual Network
ToC	Table of Contents
UC	Use Case
UMAP	Uniform Manifold Approximation and Projection
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VC	Verifiable Credentials
VMs	Virtual Machines
WMS/WFS	Web Mapping Service/ Web Feature Service
WP	Work Package
XML	eXtensible Markup Language
YALM	Yet Another Markup Language
ZIP	Zone Improvement Plan

Executive Summary

The document is the first deliverable of WP5 and reports the results of T5.1 activities aimed at recommending an evaluation methodology, performance metrics, and a timetable for the integration of the SEDIMARK platform according to the rules of decentralization, trustworthiness, intelligence, data quality, and interoperability. This deliverable is important because it defines the evaluation methodology, monitoring approach, and efficiency of what is being built, as well as the system validation through real pilot demonstrations. In order to assess the framework's capabilities from various user perspectives, the developed methodology adapts multiple quality factors implemented using technical metrics.

Before delving into the core of the deliverable, the document briefly describes the vision of the SEDIMARK marketplace, in which participants will exchange assets in a secure decentralized manner. In SEDIMARK_D2.2, the architecture's components were thoroughly examined. To create the overall decentralized solution, the integration activities are based on those components and tools under a standard development framework.

All technology providers are accountable for the various modules to which they are assigned based on a top-down integration plan that is outlined in this document. Some architecture components are not included in the first version of the platform because they are part of the platform's second and final releases. The initial release focuses on enhancing the minimum functionalities required to provide a minimum viable product. The integration plan is built upon the use case scenarios defined in T2.1 and SEDIMARK_D2.1 [4] and the timeline for the execution of the scenarios. The components are integrated using Virtual Machines (VMs), docker containers, and other orchestration tools.

This deliverable also specifies a customized evaluation process as well as numerous criteria to be employed in this evaluation. The criteria comprise technical criteria tailored to each technique/module evaluated, as well as general criteria/KPIs tailored to each use case and a metrics framework based on ISO/IEC established methods for system and product quality assessment. The standardization provides the procedures with security and compatibility. The framework will begin with the establishment of a comprehensive and meaningful set of performance metrics based on the requirements and use cases of the stakeholders. Just to remind, SEDIMARK encompasses four main use cases in different sites: Mobility Digital Twin (Finland), Urban Bike Mobility Planning (Spain), Valorisation of Energy Consumption and Customer Reactions/Complaints (Greece), and Valuation and Commercialization of Water Data (France).

1 Introduction

1.1 Purpose of the document

The main purpose of the deliverable is to streamline the project objectives within the scope of WP5. More specifically, the scope is twofold; first, to define the evaluation methodology and the metrics that will be used for each use case and, second, to specify the timetable for the integration of the separate SEDIMARK components that were defined in the relevant work packages in accordance with the architecture. The goal for the first version of the SEDIMARK platform is to release a solution with the bare minimum of core functionalities, with incremental increases in functionality in subsequent versions. This document provides a high-level theoretical approach to the solution. As a result, it is intended for a limited audience, primarily project partners, to use as a reference for upcoming activities. Other stakeholders with similar interests may also find useful ideas for developing appropriate methodologies.

1.2 Relation to other work packages and tasks

This deliverable is the outcome of the work done during the first year of the project in T5.1 (Integration and Evaluation plan and methodologies). SEDIMARK_D5.1 is a very crucial deliverable because it establishes the context for the integration activities, as well as the evaluation process and methodology. The work presented in this document is linked to T2.1 (Use Case definition) and document SEDIMARK_D2.1 [4]. The defined use cases are important in WP5 because they serve as practical examples of the solutions being developed. During the pilot demonstrations, the performance of the solution will be thoroughly evaluated and monitored using a detailed evaluation framework and performance metrics. The plan for integration is associated with the separate modules of WP3 and WP4 towards the realization of the architecture of WP2. The output of SEDIMARK_D5.1 will also be used as input to the upcoming activities of the remaining tasks (T5.2, T5.3, T5.4) of WP5 for the three integrated releases of the SEDIMARK platform which will be presented in three phases (M18-Mar. 2024, M27-Dec. 2024, M36-Sep. 2025) and analysed in the deliverables SEDIMARK_D5.2 (Integrated releases of the SEDIMARK platform. First version), SEDIMARK_D5.3 (Integrated releases of the SEDIMARK platform. Second version), SEDIMARK_D5.4 (Integrated releases of the SEDIMARK platform. Final version). This gradual platform deployment allows beneficiaries to gain valuable insights into performance and make any necessary adjustments or improvements.

1.3 Structure of the document

This document is structured into 7 major chapters:

- **Chapter 1** is the current chapter and introduces the objective of the document and how it relates to the project's activities.
- **Chapter 2** presents the scope and vision of the SEDIMARK marketplace.
- **Chapter 3** analyses the development process and the tools to create the overall decentralized SEDIMARK marketplace based on the defined architecture.
- **Chapter 4** describes all the datasets used per trial site and the software components to be integrated.

- **Chapter 5** includes a detailed timetable for integrating SEDIMARK modules. The SEDIMARK platform will be delivered in three phases: the first version, the second version, and the third version. Supported scenarios and minimum core functionalities are presented in the initial version (M18-Mar. 2024).
- **Chapter 6** presents the evaluation process and methodology, as well as performance metrics for each supported scenario. For each trial site, trial definition and KPI tables are provided for evaluation purposes.
- **Chapter 7** concludes the document, summarizing the main outcomes and the future steps in alignment with the objectives and project roadmap.

2 Scope and vision of the SEDIMARK marketplace

The SEDIMARK project is developing the technological enablers to set up a secure and decentralized marketplace in which their participants (mainly Providers and Consumers) will be able to exchange assets (basically data and services) in a trustworthy manner.

The functional architecture that has been described in Deliverable SEDIMARK_D2.2 [1] presents all the functional components that enable such a trustworthy marketplace. Furthermore, the system view that has been also described in Deliverable SEDIMARK_D2.2 [1] identifies three different domains:

- The Provider.
- The Consumer.
- The Baseline Infrastructure Facilitators (BIFs).

In the SEDIMARK context, the first two are driven by the usage of a common toolbox composed of a set of software components that implement the aforementioned functional components and are subsequently integrated into that unique software artifact. The SEDIMARK marketplace's participants will leverage such a toolbox to interact among themselves, publish and discover their Offerings, and, eventually, exchange their Assets. In contrast, the BIFs domain provides all the infrastructure and systems needed to run the Marketplace.

In this respect, the decentralized nature of the SEDIMARK marketplace will be supported by the abovementioned software artifact, the so-called SEDIMARK Toolbox, that every Participant will have to deploy and through which all the SEDIMARK marketplace's interactions will be handled.

Thus, the integration efforts that will be carried out in SEDIMARK's WP5 will result in a unique, easily deployable artifact (most likely, in the form of a software container) that the SEDIMARK's participants will be able to download and install in their respective systems. Once deployed, the SEDIMARK Toolbox will offer a unique Graphical User Interface (GUI) through which the participant (independent of whether they are a Provider or a Consumer) will be able to access the marketplace. Besides the GUI, the Toolbox will also expose programmatical Application Programming Interfaces (APIs) so that its functionalities can also be accessible by third-party software.

As it has been indicated, there will not be a separate Toolbox for Consumers and Providers, but its functionalities will be available and exposed as the participant requires. However, the project will work on different flavors of the Toolbox so that they can adapt to the capacity of the system in which they have to be deployed. This way, the objective is to be able to support the use of the Edge Computing paradigm by integrating lightweight versions of the Toolbox that can be installed on devices that do not have big storage and/or computing capabilities.

In conclusion, the SEDIMARK Marketplace will be instantiated in the form of the distributed deployments of every participant's Toolbox that, relying on the services offered by the Baseline Infrastructure [1], will interact with each other to enable their respective participants to publish and discover their Offerings and negotiate the necessary agreements under which clauses the trustworthy exchange of Assets will be, finally, performed.

3 Methodology and framework of the development process

The integration activities and development process include the components and tools developed in WP3 and WP4, to create the overall decentralized SEDIMARK marketplace based on the defined architecture. A development framework for building and training models will be established, including project templates and software scaffolds, Git repositories, workload registry and AI model registry to support the development of the AI-based solutions to be delivered in T5.3 and T5.4. To speed up development, the objective is to implement common functions (e.g., runtime, communication stacks) through (docker-based) integration layers and develop CI/CD pipelines to facilitate the integration and validation. Establishing a consistent CI/CD (Continuous Integration/Continuous Delivery) process is also a priority to support the Agile-Oriented approach [20].

3.1 CI/CD explanation

Automation is a fundamental principle for DevOps success, and CI/CD is a critical component. Continuous integration and continuous delivery (or continuous deployment) are two components of CI/CD. They form a "CI/CD pipeline" (Figure 1) [19], which is a series of automated workflows that help DevOps teams reduce manual tasks:

- **Continuous integration (CI)** automatically builds, tests, and integrates code changes within a shared repository.
- **Continuous delivery (CD)** automatically delivers code changes to production-ready environments for approval. An indicative process is depicted in Figure 2 [19].
- **Continuous deployment (CD)** automatically deploys code changes to customer directly (Figure 3).

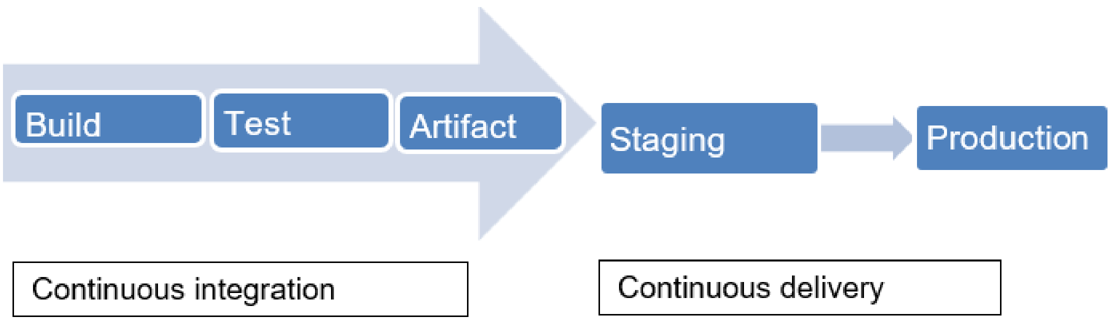

Figure 1 A CI/CD pipeline

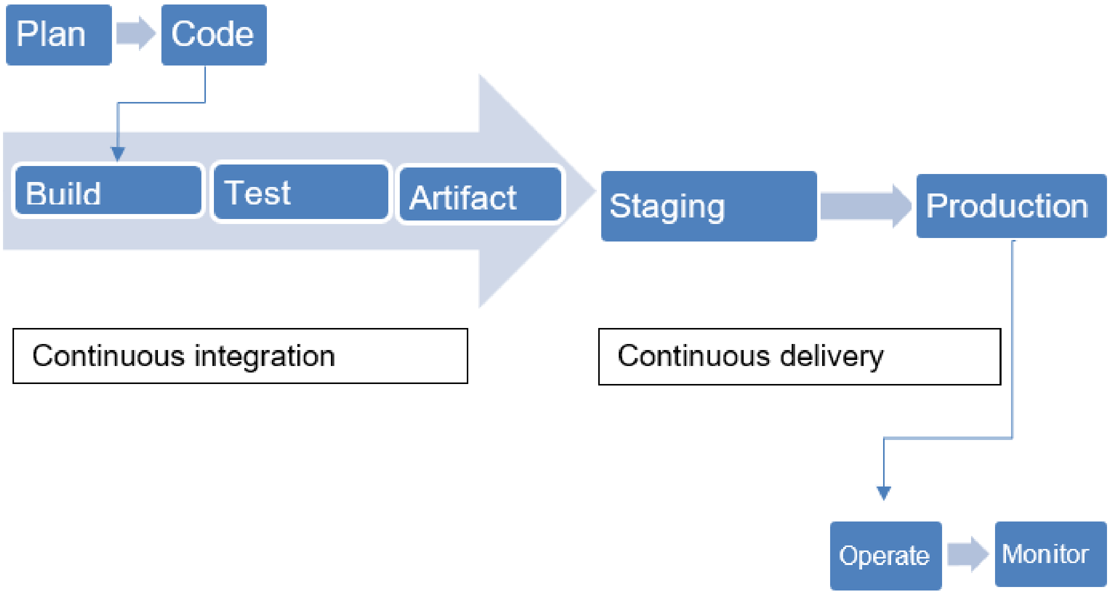

Figure 2 Continuous integration - continuous delivery

In a CI/CD pipeline that uses continuous delivery, automation pauses when developers push to production. A human still needs to manually sign off before the final release, adding more delays. On the other hand, continuous deployment automates the entire release process. Code changes are deployed to customers as soon as they pass all the required tests.

Continuous deployment is the ultimate example of DevOps automation. That doesn't mean it's the only way to do CI/CD, or the "right" way. Since continuous deployment relies on rigorous testing tools and a mature testing culture, most software processes start with CD and integrate more automated testing over time.

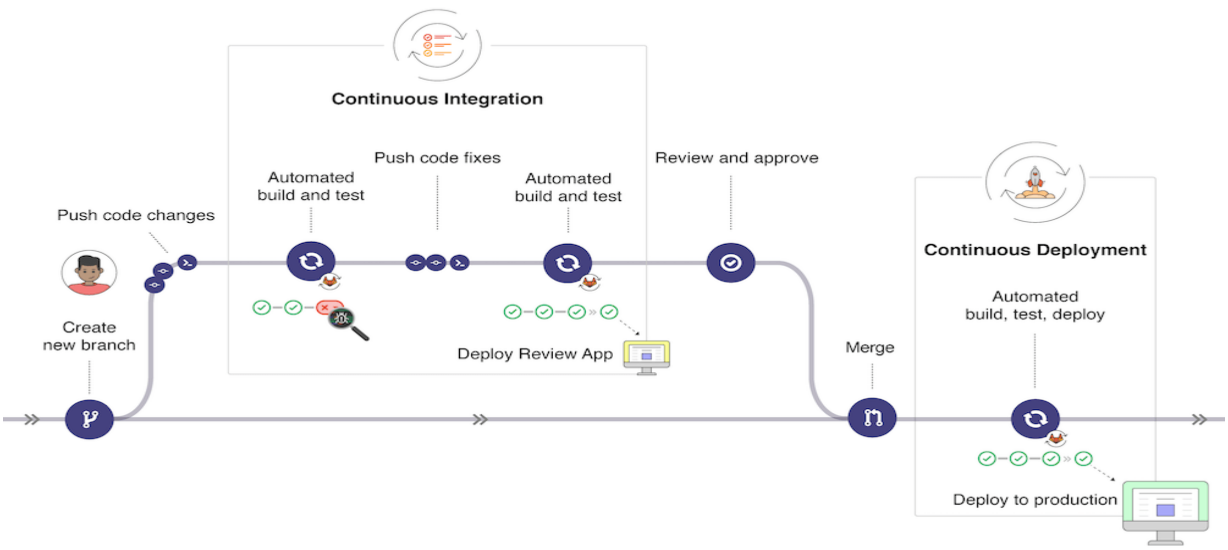

Figure 3 Continuous integration - continuous deployment

3.2 Development tools

3.2.1 GitHub actions

CI/CD refers to the process of frequently pushing code changes into the main branch while ensuring that they do not impact any changes made by others working concurrently and allows simple and convenient management of the codebase.

GitHub Actions [14] main functionalities are to:

- Simplify CI/CD by automating tasks within the software development life cycle.
- Be event-driven, i.e., we can run a series of commands after a specified event has occurred.
- Be used to automatically run software testing scripts. An event automatically triggers the workflow, which contains a job. The job then uses steps to control the order in which actions are run. The workflow is an automated procedure that is added to the repository. Workflows are one or more jobs that can be scheduled or triggered by an event, and we can use the workflow to build, test, package, release, or deploy a project on GitHub. A job is a set of steps that are executed on the same runner. By default, a workflow with multiple jobs will run those jobs in parallel.

3.2.2 Jenkins

Jenkins [15] is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. It can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed. Jenkins offers a simple way to set up a continuous integration or continuous delivery (CI/CD) environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks. While Jenkins doesn't eliminate the need to create scripts for individual steps, gives a faster and more robust way to integrate the entire chain of build, test, and deployment tools.

3.2.3 Self-hosted runners

A self-hosted runner [16] is a system that is deployed and managed to run GitHub Actions jobs. Self-hosted runners provide greater control over the hardware, operating system, and software tools than GitHub-hosted runners. Custom hardware configurations can be created with self-hosted runners to meet the needs for processing power or memory to run larger jobs, install software available on the local network, and select an operating system not offered by GitHub-hosted runners. Runners that are self-hosted can be physical, virtual, in a container, on-premises, or in the cloud.

Self-hosted runners can be placed at various levels in the management hierarchy:

- Repository-level runners are dedicated to a single repository.
- Organization-level runners can process jobs for multiple repositories in an organization.
- Enterprise-level runners can be assigned to multiple organizations in an enterprise account.

Runner machines are connected to GitHub using the GitHub Actions self-hosted runner application. The GitHub Actions runner application is open source. You can contribute and file issues in the runner repository. When a new version is released, the runner application automatically updates itself when a job is assigned to the runner, or within a week of release if the runner hasn't been assigned any jobs. A self-hosted runner is automatically removed from GitHub if it has not connected to GitHub Actions for more than 14 days. An ephemeral self-hosted runner is automatically removed from GitHub if it has not connected to GitHub Actions for more than 1 day.

3.2.4 Container registry

A container registry [17] is a repository, or collection of repositories, used to store and access container images. Container registries can support container-based application development, often as part of DevOps processes. Container registries can connect directly to container orchestration platforms like Docker and Kubernetes. Container registries can save valuable time in the creation and delivery of cloud-native applications, acting as the intermediary for sharing container images between systems.

A container image contains all the files and components that comprise an application. Containers, contrary to virtual machines (VMs), are lightweight software packages that run on top of the Linux operating system (OS). As workloads change, container images can be multiplied to scale. They are frequently linked to agile development, DevOps methodology, and continuous integration and delivery (CI/CD). Container images include system libraries, system tools, and other platform settings that applications require to run, providing developers with the portability and agility they need to quickly expand on or create new apps.

It is necessary to save, share, and access container images as they are created when developing them. A container registry essentially serves as a repository for developers to store container images and distribute them by uploading (pushing) to the registry and downloading (pulling) into another system, such as Kubernetes. Registries store application programming interface (API) paths and access control parameters for container-to-container communication, in addition to container images. APIs aid in the elimination of unintended coupling, which limits change and is a common source of outages, particularly in hybrid cloud environments where applications are no longer accommodated in the same data center. Container images can also communicate with one another through a service mesh, which is an infrastructure layer between containerized services that facilitates scaling. For cloud-native apps built in a microservices architecture, a service mesh is a way to comprise many discrete services into a functional application.

3.3 MLOps

MLOps, also known as ML Ops, is a paradigm for reliably and efficiently deploying and maintaining machine learning models in production [2]. The term is a combination of "machine learning" and the continuous software development practice of DevOps. In isolated experimental systems, machine learning models are tested and developed. When an algorithm is ready for deployment, MLOps moves it to production systems [3]. MLOps seeks to increase automation and improve the quality of production models, like DevOps or DataOps approaches, while also focusing on business and regulatory requirements. While MLOps began as a collection of best practices, it is gradually evolving into a stand-alone approach to ML lifecycle management (Figure 4) [18].

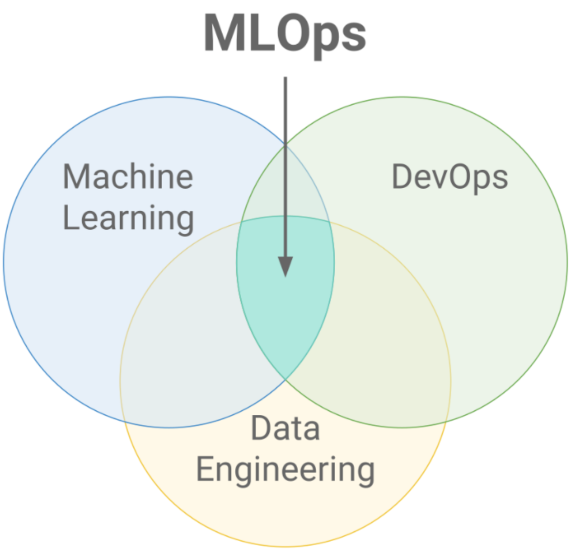

Figure 4 MLOps Venn diagram

3.3.1 Experiment tracking

Experiment tracking is the process of managing and tracking all the different machine learning experiments and their components, providing visibility into parameters, metrics, and results for better reproducibility and collaboration. As a result, it enables us to:

- **Organize** all the necessary components of a specific experiment. It's important to have everything in one place and know where it is so you can use it later.
- **Reproduce** past results easily using saved experiments.
- **Log** iterative improvements across time, data, ideas, teams, etc.

There are many options for experiment tracking but in SEDIMARK MLFlow (100% free and open source) it is going to be used because it has all the functionality needed. We can run MLFlow on our own servers and databases so there are no storage costs/limitations, making it one of the most popular options. There are also several popular options such as a Comet ML, Neptune, Weights and Biases [21-23]. These are fully managed solutions that provide features like dashboards, reports, etc.

3.3.2 Model registry

A model registry is a repository used to store and version trained machine learning (ML) models. Model registries greatly simplify the task of tracking models as they move through the ML lifecycle, from training to production deployments and ultimately retirement. In addition to the models themselves, a model registry stores information (metadata) about the data and training jobs used to create the model. Tracking these requisite inputs is essential to establish lineage for ML models. In this way, a model registry serves a function analogous to version control systems (e.g., Git, SVN) and artifact repositories (e.g., Artifactory, PyPI) for traditional software.

Each model in a model registry is given a unique identifier, which is also known as a model ID or UUID. Many commercially available registry tools include a mechanism for tracking multiple versions of the same model. The model ID and version can be used by data science and machine learning teams to refer to specific models for comparison and deployment confidence. Registry tools can also store parameters or metrics. When registering a model, for example, training and evaluation jobs could write hyperparameter values and performance metrics (such as accuracy). Storing these values allows for easy model comparison. Having this data on hand can help teams see if new versions of a model improve on previous versions as they develop new models. Many registry tools also include a graphical interface to visualize these parameters and metrics.

Model registries are generally comprised of the following elements:

- Object storage (such as Amazon S3 or Azure Blob Storage) to hold model artifacts and large binary files.
- A structured or semi-structured database to store model metadata.
- A graphical user interface (GUI) that can be used to inspect and compare trained models.
- A programmatic API that can be used to retrieve model artifacts and metadata by specifying a model ID or query.

3.3.3 Model serving

Developing a model is one thing; serving a model in production is quite another. When a data scientist has finished developing a model, the next step is to deploy it so that it can serve the application. There are two types of models serving in general: batch and online. Batch means that you feed the model with a large amount of data, typically as a scheduled job, and write the output to a table. Online deployment entails deploying the model with an endpoint so that applications can send requests to the model and receive a quick response with low latency.

The basic meaning of model serving is to host machine-learning models (on the cloud or on premises) and to make their functions available via API so that applications can incorporate AI into their systems. Model serving is crucial, as a business cannot offer AI products to a large user base without making its product accessible. Deploying a machine-learning model in production also involves resource management and model monitoring including operations stats as well as model drifts.

A deployed model is the result of any machine-learning application. Some necessitate simple deployments, while others necessitate more complex pipelines. Amazon, Microsoft, Google, and IBM all offer tools that make it easier to deploy machine-learning models as web services. Furthermore, advanced tools can automate time-consuming workflows for developing machine-learning model services.

A monolithic system may embed a machine-learning model and not expose the model available outside the system. This type of architecture requires every application using the same machine-learning model to own a copy. If there are many such applications, it quickly becomes a nightmare for MLOps. A better approach is to make the machine-learning model accessible to multiple applications via API. This deployment type has various names, including model serving, ML model serving, or machine learning serving, but they all mean the same thing.

Model serving, at a minimum, makes machine-learning models available via API. A production-grade API has the following extra functions:

- **Access points (endpoints)**: An endpoint is a URL that allows applications to communicate with the target service via HTTPS protocol.
- **Traffic management**: Requests at an endpoint go through various routes, depending on the destination service. Traffic management may also deploy a load-balancing feature to process requests concurrently.
- **Pre- and post-processing requests**: A service may need to transform request messages into the format suitable for the target model and convert response messages into the format required by client applications. Often, serverless functions can handle such transformations.
- **Monitor model drifts**: We must monitor how each machine-learning model performs and detect when the performance deteriorates and requires retraining.

3.3.4 Model monitoring

The lifecycle of machine learning doesn't stop the moment a model is deployed. Model performance monitoring is a basic operational task that is implemented after an AI model has been deployed. ML teams need a strategy to quickly adapt ML models to the constantly changing patterns in real-world data.

The tracking of an ML model's performance in production is known as machine learning model monitoring. Monitoring machine learning models is a critical feedback loop in any MLOps system for keeping deployed models current and predicting accurately, and ultimately ensuring they deliver long-term value. When live models encounter data that is significantly different from the training data, previous data becomes obsolete.

ML models in production, by definition, make inferences on constantly changing data. Even models trained on massive data sets with meticulously labelled data begin to degrade over time due to concept drift. Changes in the live environment, such as shifting behavioral patterns, seasonal shifts, new regulatory environments, market volatility, and so on, can have a significant impact on a trained model's ability to predict accurately. Without dedicated model monitoring best practices, ML and business teams have no way of knowing when the predictive performance of a model is starting to decline. If drift occurs without detection, businesses can be exposed to serious risks and erode end user trust in customer-facing applications.

To protect the value of AI applications, ML teams need to implement a system for early and proactive detection of deviations, without having to monitor models manually or build additional tooling in-house. There are several tools on the market that offer prebuilt monitoring capabilities that do not require coding, making them ideal for a team with diverse skill sets. The features below are important to look out for:

- **Built-in Model Monitoring**: The simplest way to implement model monitoring across the organization is to use a system that is natively built-in to the existing MLOps platform. This allows anyone on the team to monitor any model in one centralized dashboard.
- **Automated Retraining**: Automating the entire training pipeline, including all relevant steps in the pipeline, can save teams lots of time. The output is a production-ready model that is ready to be deployed.
- **Automated Drift Detection**: The core function of any monitoring solution. Even with hundreds of models running simultaneously, a drift-aware system will automatically

detect drifting, anomalies, data skew, and model drift. It's important to note that if your use case includes streaming data, the monitoring system needs to support real-time detection.

- **Feature Store Integration**: Maintain consistency between projects and improve collaboration across teams by using a feature store. Feature vectors and labels can be stored and analysed in the feature store, and then easily compared to the trained features and labels running as part of the model development phase.

3.3.5 Data versioning

Versioning refers to the process of uniquely naming multiple iterations of an ML model used at different stages of ML development. It helps track and control all changes applied to various versions allowing the easy recovery of a previous model version when needed.

ML experiment involves different project versions with specific enhancements or changes in each version. These changes might include:

- Update features.
- Update parameters.
- Adjust parameters.
- Add the new dataset and features.
- Readjust parameters.

Data versioning tools allow:

- **Capturing** the versions of data and models and switching between different versions as needed. It offers a unified way of accessing data, code, and ML models.
- **Reproducibility**: ML versioning aids in the finalization of the best model and its trade-offs. It is critical for ensuring reproducibility in ML experiments. By capturing a snapshot of the entire ML pipeline, it is possible to reproduce the same results while saving time and effort on retraining and testing.
- **Better tracking**: ML workflows are error-prone and complex and hence require tracking. ML models can fail or underperform due to a variety of factors such as adding more data or updating features. Model versioning enables the reversion of failed models to previous, stable, and working versions.
- **Track dependencies**: ML experiments involve complex workflows with several variables that influence model performance. Datasets, frameworks, feature sets, and test cases, for example, all contribute to model performance. Model versioning aids in the tracking of dependencies that affect the performance of ML models. It helps with the testing of multiple models in various ML pipelines, tuning parameters and hyperparameters, and maintaining model accuracy.
- **Scaled AI-ML governance**: ML projects are rolled out iteratively for scaled performance and failure tolerance. Model versioning supports better AI governance with access control, policies, the right version deployments, and model activity tracking.

4 Datasets and software components to be integrated

In this section, all the software components that were introduced in WP3 and WP4 will be analysed with the same template, as well as the datasets used.

4.1 Reminder of SEDIMARK architecture

The complete description of the SEDIMARK architecture is detailed in SEDIMARK Deliverable SEDIMARK_D2.2, so the reader is referred to that document for having the full picture of the project's functional and system architectures. Here, in this section, we will include a summary of the architecture for completeness.

SEDIMARK aims to provide a fully decentralized secure and intelligent data and services marketplace, where providers and consumers can exchange their assets in a trustworthy manner and build knowledge and intelligence upon them. A fully decentralized solution means that there is no central point of control or central point that gathers all data, services, assets, etc. but participants exploit the decentralized nature of DLT to connect directly to each other and exchange assets in a secure way, allowing the asset providers to keep their assets locally and have full control over who gets access to their data, when and how.

The high-level functional view of the SEDIMARK architecture is depicted in Figure 5 below, showing the splitting of the architecture into six architectural layers, each one consisting of various functional modules that perform the main functions related to this layer:

- **Data layer** includes all the functionalities for processing, curating, formatting, annotating and improving the quality of data(sets).
- **Intelligence layer** includes all functionalities to build ML/AI models on top of the processed datasets, i.e., training models locally and distributedly, optimizing models, changing model formats, providing inference, building analytics, etc.
- **Interaction layer** enables the connectivity of the nodes providing the functionalities to connect to the DLT.
- **Services layer** includes the main functionalities for managing the services provided within SEDIMARK, including a user interface, the registration, discovery and sharing of offerings, recommendations, payments, contracting, etc.
- **Distributed storage layer** manages the local and distributed storage facilities.
- **Trust layer** includes functionalities to build trust in the decentralized architecture, i.e., decentralized identities, verifiable credentials, data integrity, etc.

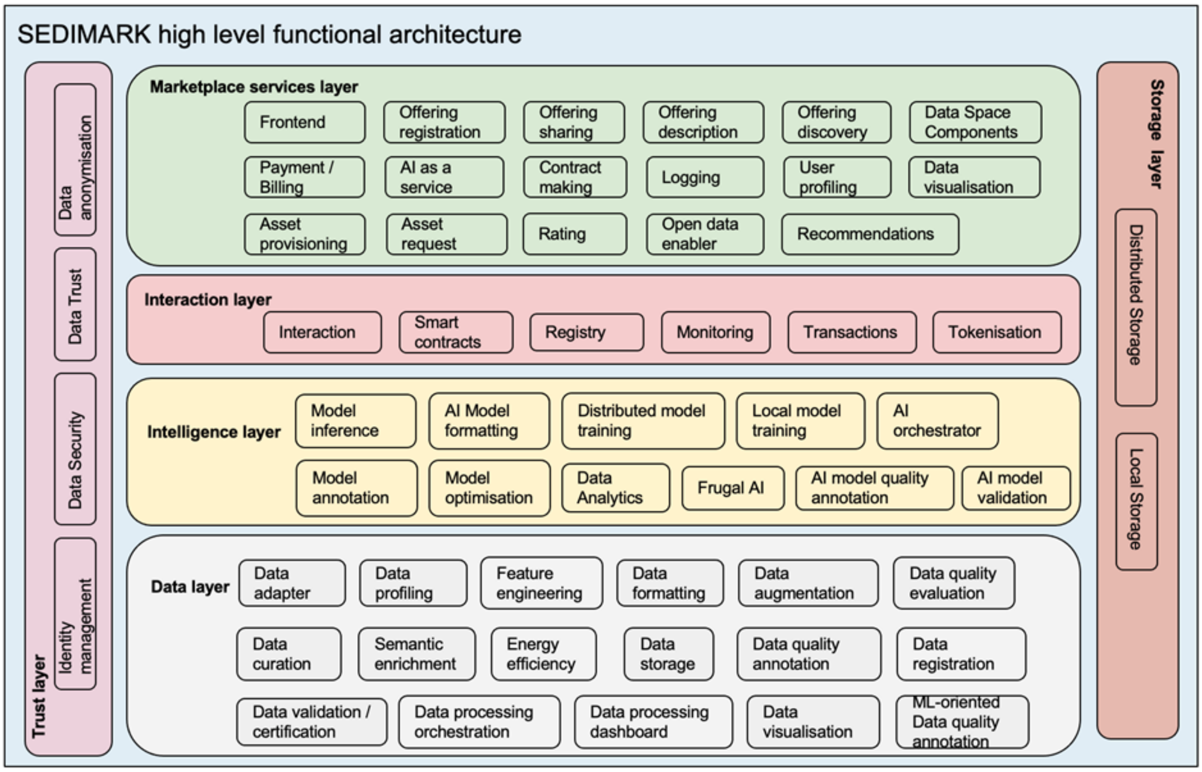

Figure 5 Functional view of the SEDIMARK architecture

The SEDIMARK architecture assumes that there are two main user roles:

- Providers, who are the ones providing the assets for sharing through the marketplace, with the assets being datasets, services, AI models, etc.
- Consumers, who are the ones who are consuming the assets that are being shared.

It is assumed that there will be different instantiations of the functional toolbox of SEDIMARK based on the role of each participant to cater for the different functionalities that each role will utilize. For example, data processing functionalities might only be used by the Providers to clean their data and improve their quality before they are shared in the marketplace while offering discovery, recommendations, and even most ML-related functions might only be used by Consumers.

4.2 Datasets per trial site

4.2.1 Datasets from Mobility Digital Twin in Helsinki

The data pertaining to urban mobility can be divided into three categories:

- Data describing the infrastructure.
- Data depicting mobility events.
- Data describing environmental and other conditions.

The datasets may be:

- Static, e.g., archived datasets, maps, statistics etc.
- Dynamic, e.g., areas with occasional changes in e.g., size and other parameters.

- Real-time or near-real-time, e.g., traffic measurements such as volumes, speeds, routes, travel times.

The data will be in a multitude of formats. Helsinki is committed to using open standards, but some of the data (e.g., maintenance-related) may not yet have established standards for the mobility domain and may be experimental.

A big share (>50%) of the data is spatial data in some form, whether describing infrastructure or certain parameters of public space, or measurements or forecasts with a spatial component.

A full description of the detailed dataset information can be found in the Annex.

4.2.2 Datasets from Urban bike mobility planning in Santander

Santander City Council has set up the Santander Smart City Platform (SSCP), which brings together all the operating data of the municipal services in a single centralized point which, in turn, provides multiple information services, both to the municipal departments themselves and to other interested parties. It also has facilities for integrating information into dashboards and customized reports.

The Marketplace will have a direct relationship with the SSCP in such a way that it will be fed both by existing data and data that may be collected during the project, while at the same time, it will be able to absorb information from other Marketplace actors that may be useful for municipal departments and, in general, for all SSCP stakeholders.

The City Council, as a public administration, aims for efficiency in the use of resources and transparency in management. The Marketplace adds a new aspect, contemplated in the municipal strategy, which is to help energize the productive fabric of its environment in line with European and national guidelines in relation to the data economy.

The functional requirements will be similar to those described in other use cases. However, it is important to add an element due to the interaction that the Marketplace is going to have with the SSCP where the data model is NGSI v2, which has implications when implementing interoperability.

The data sources to be integrated have been described in Deliverable SEDIMARK_D2.1 [4]. In this section, the most important ones related to bicycle mobility in the urban environment are indicated:

- Data on the new municipal electric bicycle rental service (scheduled to be launched in 2024).
- Data on the availability of the current bicycle rental system.
- Data on the use of covered bike racks.
- Data on the new devices developed under the project.
- Data from the bicycle counting sensors that count the number of bicycles circulating in some lanes of the city.

More details about the datasets can be found in the Annex.

4.2.3 Datasets from Valorisation of energy consumption and customer reactions/complaints in Greece

The dataset for the use case of “Valorisation of Energy Consumption and Customer Reactions / Complaints in Greece” led by MYT contains the following:

- Numerical data (including but not limited to consumption values, residential size in square meters, supply IDs, tax IDs).
- Fields with text (string).
- Weather data (numerical such as temperature and humidity).
- Wind direction (mixture of textual and numerical data).

The structure of the dataset is a column-based "csv" file. On average, this "csv" file contains 8 columns of data for 5000 unique customers. The data is initially stored in MYT corporate data warehouse.

Energy-oriented data for the energy consumption prediction and clustering will be public and anonymized except for the ZIP Codes. This means that the only data that will have public status are the weather data, the residential size and any residential consumption related data. Customer-oriented data regarding segmentation and churn prediction will be private and anonymised. As such, not all data will be shared in the marketplace, but all data processed will concern a specific time range.

Following the data cleaning task, the Machine Learning analytical tasks that will most likely be performed are the model training, the testing and the data validation process. SEDIMARK tools are expected to run on MYT data servers.

More details about the datasets can be found in the Annex.

4.2.4 Datasets from Valuation and commercialization of water data in France

The water use case will include different datasets:

- Meteorological data from open API [5].
- Hydrometric measurements data from open API [6].
- Measurements from on-site sensors.

The data from the API are retrieved by an Stellio context broker and available on its API in NGSI-LD. The data from the sensors are all simple time series (fields: datetime and numeric value), updated in real time (not static) with sometimes contextual metadata (height of measurement, source, etc.).

The volume of data would be one measurement per hour, per parameter which gives an order of approximately 500 daily values for around 20 parameters (meteorological - hydrometric - sensors).

The data will be shared on the SEDIMARK marketplace. We expect to run outlier detection as data cleaning tasks, and probably interpolation of missing data for more general data processing. Machine learning analytics will also be performed on the data to have a forecast of some parameters. We expect to run SEDIMARK tools on edge devices.

4.3 Software components

Based on the functional view of the SEDIMARK architecture depicted in Section 4.1, a detailed description of the components that are part of the first version (release on M18-Mar. 2024) is presented. There are still some components that exist in the architecture, but they are not in the list below (since they are part of the second and third integrated versions). For these components, there are assignments per component and partner proposed by UCD.

For homogeneity purposes, each component will be described using the following template (Table 1) which contains information about inputs/outputs, methods and datasets, language, modules, timetable, etc.

Table 1 Template for component description

Field	Title of software component
Overview	Introduce the new software in a few lines. Describe what the product will do and what problems it will solve.
Responsible partner	Partner responsible for the design and the implementation.
Inputs	Describe the inputs and from whom (user/other component) the inputs will come.
Outputs	Describe the outputs and to whom (user/other component) the outputs will be given.
Methods used	Methods/techniques used for the implementation.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Software language/framework used for the development or needed for the execution of the software component.
Modules	Software modules used for the development of the software component, e.g., libraries such as pytorch, tensorflow.
Deployment	Describe how the software component will be deployed, e.g., as a Docker container.
Development timeline	What will be ready in M18-Mar. 2024 (SEDIMARK_D5.2), M27-Dec. 2024 (SEDIMARK_D5.3), M36-Sep. 2025 (SEDIMARK_D5.4).
Assumptions, dependencies, and constraints	Any assumptions, dependencies, and constraints.
Use case relevance	Relevance to 1 or more SEDIMARK use cases.
Additional documentation	Reference to any documentation if it exists.

The components in the subsections below will be described layer by layer starting from the bottom of the architecture (data layer) and levelling up (marketplace services layer). Some components may be described with "sub-components" for better understanding.

4.3.1 Data layer

4.3.1.1 Data adapter

Table 2 Data adapter description

Field	Data Adapter
Overview	The data adapter has the responsibility to translate the data in the format of the data platform to the one internal to the SEDIMARK processing model.
Responsible partner	EGM
Inputs	The official data adapter will be made to translate data stored in an NGSI-LD context broker.
Outputs	The requested data is in the internal format for processing (based on Pandas data frames and Python dictionaries).
Methods used	NGSI-LD query language
Datasets used	Should be usable with as many datasets as possible.
Language	Python
Modules	Pandas, NGSI-LD library [29]
Deployment	As a processing step in a SEDIMARK processing pipeline.
Development timeline	• M18 (Mar. 2024): First simple and general implementation to validate the concept. • M27 (Dec. 2024): Adapted to different data models. • M36 (Sep. 2025): Final implementation / tested.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.1.2 Semantic enrichment

Table 3 Semantic enrichment description

Field	Semantic Enrichment
Overview	The component will be responsible for semantically enriching data assets based on results generated from the data processing pipeline.
Responsible partner	INRIA, SURREY, SIE, EGM, UC
Inputs	Generated analytical and contextual abstractions from intermediate data processing pipeline sub-components.
Outputs	Semantically annotated description that will be appended to offered (final) data assets.
Methods used	Translation, formatting, linking
Datasets used	All
Language	Python, Java
Modules	ngsild-client [29], PyLD [30], pandas, rdflib-jsonld [31], Apache Jena [32]
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): Enrich data assets with data analytical context. • M27-Dec. 2024 (SEDIMARK_D5.3): Enrich data assets with domain-specific knowledge. • M36-Sep. 2025 (SEDIMARK_D5.4): Final implementation.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.1.3 Data processing orchestration

Table 4 Data processing orchestration description

Field	Data processing orchestration
Overview	Responsible for the orchestration of the data processing pipeline
Responsible partner	SURREY, SIE
Inputs	The input will originate from the data adaptor.
Outputs	The output will be forwarded to intermediate data processing components.
Methods used	DAG [33]
Datasets used	All data assets
Language	Python
Modules	Airflow, Mage.ai
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): Concept validation. • M27-Dec. 2024 (SEDIMARK_D5.3): Address Provider/Consumer usage feedback. • M36-Sep. 2025 (SEDIMARK_D5.4): Final implementation.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

Table 5 Data processing management description

Field	Data processing management
Overview	Manages the various processing pipelines. Allows the users to define and configure new processing pipelines and trigger them when needed
Responsible partner	EGM, SURREY, SIE
Inputs	Configuration files and/or GUI

Field	Data processing management
Outputs	Processing pipelines configured and ready to be triggered in the data platform
Methods used	Object-oriented development
Datasets used	All
Language	Python
Modules	n/a
Deployment	n/a
Development timeline	• M18 (Mar. 2024): very rough interface (only config files probably), basic functionalities. • M27 (Dec. 2024): more advanced functionalities, GUI. • M36 (Sep. 2025): implementation finalized and tested.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.1.4 Data formatting

Table 6 Data formatting description

Field	Data formatting
Overview	Data formatting component will translate the data expressed in various formats provided by providers into the SEDIMARK format. The NGSI-LD format is the one adopted within SEDIMARK, which will make the heterogeneous data easier to process mainly within the Data processing enabler and in interaction with the AI enabler.
Responsible partner	INRIA, EGM, SIE
Inputs	A dataset or a data stream in the Provider's format.
Outputs	A dataset or a data stream in the SEDIMARK format.

Field	Data formatting
Methods used	Methods for data conversion from one format to another.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2
Language	Python
Modules	n/a
Deployment	n/a
Development timeline	• M18 (Mar. 2024): first general implementation. • M27 (Dec. 2024): advanced and adapted version that includes different providers' formats.
Assumptions, dependencies, and constraints	AI enabler, Data processing enabler
Use case relevance	All
Additional documentation	n/a

4.3.1.5 Data curation

Table 7 Data curation description

Field	Data curation
Overview	The data curation enabler includes functionalities for data profiling, anomaly/outlier/noise detection, duplicate detection and missing value imputation and is used in conjunction with the data processing pipeline in order to assess and improve the data quality.
Responsible partner	UCD, INRIA, UC
Inputs	The input is the dataset for curation or a single data point in the case of streaming data.
Outputs	The output is an annotated dataset (in internal format) with extra fields for the quality metrics and either additional fields for flagging the "low quality" entries or with the bad entries removed.

Field	Data curation
Methods used	Anomaly detection, missing value imputation, deduplication, data profiling
Datasets used	Any dataset
Language	Python
Modules	Pyod, pycaret, sklearn, pandas, tods, etc.
Deployment	As a processing step in a SEDIMARK processing pipeline.
Development timeline	• M11 (Aug. 2023): First prototype. • M15 (Dec. 2023): Improved version. • M18 (Mar. 2024): First Integration with other components. • M27 (Dec. 2024): Second improved version integrated. • M36 (Sep. 2025): Final implementation / tested.
Assumptions, dependencies, and constraints	• Required additional user input for the various components to work efficiently. • Requires user input for the type of processing/curation to be done.
Use case relevance	All
Additional documentation	n/a

4.3.1.6 Feature engineering

Table 8 Feature engineering description

Field	Feature engineering
Overview	Feature engineering refers to the preprocessing steps that select and transform features to simplify and speed up data transformations while enhancing model accuracy.
Responsible partner	INRIA, SIE, UCD
Inputs	A dataset or a data stream in the SEDIMARK format.

Field	Feature engineering
Outputs	Pre-processed data containing relevant features and enhanced data features that would serve for local/distributed model training.
Methods used	Methods for feature engineering for feature selection, feature extraction, etc.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python
Modules	Sklearn, PCA, UMAP, random_projection, IncrementalPCA, etc.
Deployment	n/a
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): Working on a first version. • M27 (Dec. 2024): Advanced and adapted version.
Assumptions, dependencies, and constraints	Data augmentation, data cleaning, energy efficiency, AI model training.
Use case relevance	All
Additional documentation	n/a

4.3.2 Intelligence layer

4.3.2.1 Local model training

Table 9 Customer segmentation and churn prediction description

Field	Customer segmentation and churn prediction
Overview	This module's objective is to segment our customer base and forecast churn propensity, culminating in a calculated churn probability for each individual customer.
Responsible partner	WINGS
Inputs	Dataset about the characteristics of the customer.

Field	Customer segmentation and churn prediction
Outputs	Classification of a customer into a category and churn prediction.
Methods used	LightGBM, XGBoost, Catboost
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python
Modules	LightGBM, XGBoost, Catboost [34]
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (First integrated version- SEDIMARK_D5.2). • M27-Dec. 2024 (Second improved version). • M36-Sep. 2025 (Final implementation).
Assumptions, dependencies, and constraints	n/a
Use case relevance	Relevance to 1 or more SEDIMARK use cases.
Additional documentation	n/a

4.3.2.2 Data analytics

Table 10 Energy consumption prediction description

Field	Energy consumption prediction
Overview	The electricity consumption prediction module harnesses a week's worth of time-series energy consumption data, preceding the decision-making juncture, to forecast subsequent daily consumption in hourly intervals.
Responsible partner	WINGS
Inputs	Electricity consumption values in the form of time series.
Outputs	Electricity consumption prediction in the form of time series.

Field	Energy consumption prediction
Methods used	DeepAR
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python
Modules	GluonTS [35]
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (First integrated version- SEDIMARK_D5.2). • M27-Dec. 2024 (Second improved version). • M36-Sep. 2025 (Final implementation).
Assumptions, dependencies, and constraints	n/a
Use case relevance	Relevance to 1 or more SEDIMARK use cases.
Additional documentation	n/a

4.3.2.3 Distributed model training

Table 11 Classic federated learning in multi-party computation scheme description

Field	Classic federated learning in multi-party computation scheme
Overview	The aim is to build a framework focused on distributed learning by utilizing established federated learning techniques. At the core of this strategy is the implementation of a multi-party computation system. This system is specially engineered to eliminate the necessity for a central server, thereby safeguarding the secure calculation of necessary updates to model parameters.
Responsible partner	WINGS
Inputs	Model architecture, local dataset
Outputs	Global model

Field	Classic federated learning in multi-party computation scheme
Methods used	Federated averaging (Classic FL), Multi-party computation (MPC)
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python
Modules	Fleviden, shamrock.AI
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (First integrated version- SEDIMARK_D5.2). • M27-Dec. 2024 (Second improved version). • M36-Sep. 2025 (Final implementation).
Assumptions, dependencies, and constraints	n/a
Use case relevance	Relevance to 1 or more SEDIMARK use cases.
Additional documentation	n/a

Table 12 Meta-learning of ensemble model weights description

Field	Meta-learning of ensemble model weights
Overview	A sophisticated framework enabling the aggregation of an ensemble model from a diverse set of individual models present in a distributed network.
Responsible partner	WINGS
Inputs	Local models, datasets
Outputs	Global model
Methods used	Ensemble learning
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python

Modules	n/a
Deployment	Docker container
Development timeline	• M18-Mar. 2024 (First integrated version- SEDIMARK_D5.2). • M27-Dec. 2024 (Second improved version). • M36-Sep. 2025 (Final implementation).
Assumptions, dependencies, and constraints	n/a
Use case relevance	Relevance to 1 or more SEDIMARK use cases.
Additional documentation	n/a

Table 13 Service-shared federated learning with Fleviden description

Field	Service-shared federated learning with fleviden
Overview	This component uses two background assets for the federated learning process: fleviden and fleviscript. The fleviden tool is a collection of small software components called pods that can be assembled in different ways to build complex distributed computing programs and federated learning protocols. The fleviscript tool is a language and interpreter to define the way fleviden components are connected.
Responsible partner	ATOS, WINGS
Inputs	The input is the model to be trained and the dataset to be used. The user provides: • A fleviscript program. • An initial model definition in Keras v3 format.
Outputs	The output is a fully trained version of the model. The user obtains: • A global trained model in Keras v3 format.

Field	Service-shared federated learning with fleviden
Methods used	Federated Learning (fleviden)
Datasets used	Any dataset
Language	Fleviden
Modules	The fleviden and fleviscript tools.
Deployment	• As a step in a SEDIMARK AI pipeline. • Docker container containing a REST API to push and execute incoming fleviscripts / model programs.
Development timeline	• M12 (Sept. 2023): First prototype. • M15 (Dec. 2023): Improved version. • M18 (Mar. 2024): First Integration with other components. • M27 (Dec. 2024): Second improved version integrated. • M36 (Sep. 2025): Final implementation / tested.
Assumptions, dependencies, and constraints	Required additional user input for the model to be trained, the dataset used and the type of distributed training to be done.
Use case relevance	All
Additional documentation	To be provided on a per-request basis.

Table 14 Model-shared federated learning with shamrock.AI description

Field	Model-shared federated learning with sha mrock.AI
Overview	This model training component allows users to train ML models in a distributed or decentralized manner using Federated Learning or Gossip Learning approaches.
Responsible partner	UCD, WINGS

Field	Model-shared federated learning with sha mrock.AI
Inputs	The input is the model to be trained in a common description format including the model parameters and the dataset to be used.
Outputs	The output is a fully trained version of the model.
Methods used	Federated Learning, Gossip Learning, Distributed Reinforcement Learning
Datasets used	Any dataset
Language	Python
Modules	shamrock.AI
Deployment	As a step in a SEDIMARK AI pipeline
Development timeline	• M13 (Oct. 2023): First prototype. • M15 (Dec. 2023): Improved version. • M18 (Mar. 2024): First Integration with other components. • M27 (Dec. 2024): Second improved version integrated. • M36 (Sep. 2025): Final implementation / tested.
Assumptions, dependencies, and constraints	Required additional user input for the model to be trained, the model parameters, the configuration of the training process, the dataset used and the type of distributed training to be done. Depends on the authorization/access control module to restrict access to the training process to non-authorized users.
Use case relevance	All
Additional documentation	n/a

4.3.2.4 AI orchestrator

Table 15 AI orchestrator description

Field	AI Orchestrator
Overview	Orchestrate a set of AI / ML models so that they can work together, either in parallel, concurrently or as a sequence of processing.
Responsible partner	EGM
Inputs	A dataset or a streamed data flow and a description of the AI / ML models to be executed.
Outputs	The results of the processing of some or all the AI / ML models.
Methods used	• Design of format to describe the orchestration of a set of AI / ML models. • Design of rules and formats to ensure and enforce the interoperability of the selected AI / ML models. • Development of a runtime orchestrator engine to execute the defined orchestration graph.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python
Modules	n/a
Deployment	Docker container

Field	AI Orchestrator
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): Experimental version using some datasets used by the urban bike mobility planning in Santander. • M27-Dec. 2024 (SEDIMARK_D5.3): Validated version supporting simple use-cases making use of parallel, concurrent, and sequential processing chains. Runtime monitoring of the execution flow. • M36-Sep. 2025 (SEDIMARK_D5.4): Extended version supporting more complex use-cases.
Assumptions, dependencies, and constraints	• Orchestrated AI / ML models must expose an API using a common data format. • Orchestrated AI / ML models must publish or expose some metadata to allow for an easier setup of the orchestration.
Use case relevance	Relevance to the urban bike mobility planning use-case in Santander.
Additional documentation	n/a

4.3.2.5 Model annotation

Table 16 Model annotation description

Field	Model annotation
Overview	The annotation is composed of: • AI model quality annotation that consists of the annotation of AI models depending on their corresponding performance and accuracy. • Data quality annotation that adds information to data based on a set of quality metrics. • ML-oriented data quality annotation that reveals if the data is of good quality and could be used to train ML algorithms. • Semantic annotation that aims to enhance the data quality by adding information in the form of metadata.
Responsible partner	INRIA, UCD, EGM, UC, WINGS
Inputs	• An AI model performance measure (e.g., accuracy, MSE, silhouette measure, memory, running time) obtained from a local or distributed model training and model inference. • Data quality metrics' results.
Outputs	• An AI model linked with metadata derived from its performance (quality) which can serve AI model offering description, model optimization, and offering discovery components. • Annotated data with metadata based on its quality.
Methods used	Annotation techniques
Datasets used	Any datasets used or needed from the list of datasets in section 4.2
Language	Python
Modules	Pandas, NGSI-LD Library [29]

Field	Model annotation
Deployment	n/a
Development timeline	• M18 (Mar. 2024): A first implementation. • M27 (Dec. 2024): A more adapted version. • M36 (Sep. 2025): A final version.
Assumptions, dependencies, and constraints	Local model training, Distributed model training, Data analytics.
Use case relevance	All
Additional documentation	n/a

4.3.2.6 AI model formatting

Table 17 AI model formatting description

Field	AI model formatting
Overview	Package a trained ML model into a distribution format that can be used natively to run the ML model.
Responsible partner	EGM, INRIA, SIE
Inputs	An ML model trained with a well-known ML framework (TensorFlow, Keras, etc.).
Outputs	An artefact that can be deployed and used to perform real time ML processing.
Methods used	Integration of the BentoML platform and toolkit.
Datasets used	No dependency on the datasets.
Language	Python
Modules	BentoML Python client library [36]
Deployment	Docker container

Field	AI model formatting
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): working version of the integration of the BentoML platform. • M27-Dec. 2024 (SEDIMARK_D5.3): use of the component in advanced use-cases. • M36-Sep. 2025 (SEDIMARK_D5.4): focus on the performance and scalability of the component, stabilization.
Assumptions, dependencies, and constraints	The ML model must have been trained with one of the ML frameworks supported by BentoML.
Use case relevance	All
Additional documentation	n/a

4.3.3 Interaction layer

4.3.3.1 Registry

Table 18 Registry description

Field	Registry
Overview	The registry employed is a distributed ledger, which provides trust, non-repudiable and immutable information about Participants and Offerings.
Responsible partner	LINKS
Inputs	n/a
Outputs	Underlying structure to provide Trust within the SEDIMARK domain.
Methods used	• Usage of existing open-source software for the nodes. • Customization of the configuration for SEDIMARK scenarios and constraints.
Datasets used	n/a
Language	Rust/Go [37]

Field	Registry
Modules	• Software modules for IOTA HORNET node(s). • Customized configuration for the SEDIMARK Marketplace. • Library for interacting with IOTA DLT.
Deployment	Deployment on use case Partner's physical infrastructure as a service running (DLT). Usage of Docker containers to ease the deployment and simplify the reproducibility among partners.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Assumption: Partners of the Consortium are able to provide physical infrastructure (e.g., servers, VMs, etc.) to host the services.
Use case relevance	All
Additional documentation	https://github.com/iotaledger/hornet

4.3.3.2 Interaction

Table 19 Interaction description

Field	Interaction
Overview	Software component at any Connector that enables interaction with the distributed ledger (i.e., IOTA DLT).
Responsible partner	LINKS
Inputs	n/a
Outputs	Command issuing for the interactions with the DLT.
Methods used	In-house customized software using existing open-source standard libraries.
Datasets used	n/a
Language	Rust [37]

Field	Interaction
Modules	• Communication with IOTA Ledger to issue data transactions. • Wallet to issue value transactions, to exchange value and to interact with Smart Contract.
Deployment	Deployed as a component of the SEDIMARK Toolbox.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Existing connectivity link with the registry
Use case relevance	All
Additional documentation	https://github.com/iotaledger

4.3.3.3 Smart contracts

Table 20 Smart contracts description

Field	Smart contracts
Overview	Software applications that operate on the L2 decentralized network of validators who execute and validate the same code reaching a consensus on the same valid output, providing tamper-proof code.
Responsible partner	LINKS
Inputs	Offering and User data (e.g., Wallet, Authorization Policies, etc.).
Outputs	Functional capability for trading assets.
Methods used	In-house customized software using existing open-source standard libraries.
Datasets used	n/a
Language	Rust/Solidity/TypeScript
Modules	Node(s) for IOTA Smart Contracts

Field	Smart contracts
Deployment	Deployed as an additional layer on top of the distributed ledger.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Distributed ledger is operational
Use case relevance	All
Additional documentation	https://github.com/iotaledger/wasp https://wiki.iota.org/smart-contracts/overview

4.3.3.4 Tokenization

Table 21 Tokenization description

Title	Tokenization
Overview	Software components to tokenize assets.
Responsible partner	LINKS
Inputs	Asset Offering
Outputs	NFT related to a specific Offering.
Methods used	NFT minting according to ERC-721 and ERC-20.
Datasets used	None
Language	Rust/Solidity/TypeScript
Modules	Single module
Deployment	Deployed as a component of the SEDIMARK Toolbox.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Distributed ledger is operational
Use case relevance	All

Title	Tokenization
Additional documentation	https://github.com/iotaledger/wasp

4.3.3.5 Monitoring

Table 22 Monitoring description.

Title	Monitoring
Overview	Software components able to collect the evidence from the IOTA DLT (both L1 and L2).
Responsible partner	LINKS
Inputs	Transactions on the distributed ledger.
Outputs	Log files/output
Methods used	In-house customized software using existing open-source standard libraries.
Datasets used	None
Language	Rust [37]
Modules	Single module
Deployment	Deployed as a component of the SEDIMARK Toolbox.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Distributed ledger is operational; behavior to capture must be observable.
Use case relevance	All
Additional documentation	n/a

4.3.4 Trust layer

4.3.4.1 Data trust

Table 23 Data trust description

Title	Data Trust
Overview	Provide, manage and control the aspects related to security and trust within the SEDMARK domain.
Responsible partner	LINKS
Inputs	Assets
Outputs	Trust metadata
Methods used	Creation and verification of trust metadata to be associated with the asset which is to be protected.
Datasets used	n/a
Language	Rust [37]
Modules	Software modules for implementing Data Trust.
Deployment	Deployed as a component of the SEDIMARK Toolbox
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Assumption of Trust at the data provider (however, Full Trust shall start at data source/generator). Dependencies: Data Security
Use case relevance	All
Additional documentation	n/a

4.3.4.2 Data security

Table 24 Data security description

Title	Data Security
Overview	Software components to provide cryptographic primitives for security and trust.
Responsible partner	LINKS
Inputs	Generic data (not necessarily Assets, nor Data Assets).
Outputs	Cryptographic material
Methods used	Definition and integration of secure cryptographic primitives (e.g., SHA2, SHA3, ECDSA, ChaCha20, etc.).
Datasets used	n/a
Language	Rust [37]
Modules	• Software modules for digestion. • Software modules for a/symmetric cryptography. • Software modules for digital signatures.
Deployment	Deployed as a component of the SEDIMARK Toolbox.
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Constraint: usage of secure and well-known open-source cryptographic libraries.
Use case relevance	All
Additional documentation	n/a

4.3.4.3 Identity management

Table 25 Identity management description

Title	Identity Management
Overview	Provide and manage decentralized digital identity in the SEDMARK domain.
Responsible partner	LINKS
Inputs	Elements for issuing, verifying and managing decentralized identities (VC, DID, DIDDocuments, etc.).
Outputs	Decentralized identities (VC, DID, DIDDocuments, etc.).
Methods used	Implementation to provide the decentralized Self-Sovereign Identity (SSI) model standardized by W3C.
Datasets used	n/a
Language	Rust [37]
Modules	• In-house developed software modules for Holder, Issuer and Verifier. • Library for interacting with IOTA DLT.
Deployment	• Deployment as containerized services (Issuer, Verifier). • Deployment as a software module to be integrated at the Connector (Holder).
Development timeline	• M18 (Mar. 2024): Initial version. • M27 (Dec. 2024): Intermediate version. • M36 (Sep. 2025): Final version.
Assumptions, dependencies, and constraints	Distributed ledger is operational.
Use case relevance	All
Additional documentation	https://github.com/iotaledger/identity.rs https://wiki.iota.org/identity.rs/introduction/ https://www.w3.org/TR/did-core/

4.3.4.4 Data anonymization

Table 26 Data anonymization description

Field	Data Anonymization
Overview	Anonymisation is a data privacy technique that will replace sensitive information (tax IDs, electricity supply IDs) from the dataset. By anonymizing these details, the technique addresses privacy concerns, protects customers' identities, and ensures compliance with data protection regulations. This process also reduces the risk of data breaches and unauthorized access to personal information, promoting a safer data analysis environment.
Responsible partner	MYT
Inputs	The input for the anonymization process consists of two pieces of sensitive information: • Tax ID (Tax Identification Number): A unique identification number assigned to individuals for tax purposes, often used to identify and track taxpayers. • Electricity Supply ID (Electricity Meter Number): A unique identifier associated with a customer's electricity meter, used for monitoring electricity consumption and billing purposes. The input will come from MYT.
Outputs	Anonymize all kinds of personalized inputs
Methods used	The "rank(dense)" method will be used to anonymize both inputs, assigning unique integers to each distinct value. The "dense" method ensures that no ranks are skipped, and each unique value gets a unique integer rank.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	Python

Field	Data Anonymization
Modules	Pandas
Deployment	Deployment from MYT on their servers.
Development timeline	The necessary dataset that will be anonymised.
Assumptions, dependencies, and constraints	Columns Tax ID & Supply ID should be integers.
Use case relevance	Valorisation of energy consumption and customer reactions/complaints in Greece.
Additional documentation	n/a

4.3.5 Storage layer

4.3.5.1 Distributed storage

Table 27 Distributed storage description

Field	Distributed Storage
Overview	This component will be responsible for the distributed storage of data assets.
Responsible partner	SURREY, UC, SIE
Inputs	Final data assets to be offered at the marketplace, which will originate from the data processing pipeline.
Outputs	Final data assets are stored based on Participant computing capabilities.
Methods used	Methods/techniques used for the implementation.
Datasets used	All
Language	Python
Modules	NGSI-LD Context Broker (e.g.: Stellio, Orion-LD, Scorpio…), MinIO cloud storage server.
Deployment	Docker container

Field	Distributed Storage
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2), Storage of Provider data assets on multiple stores within the provider. • M27-Dec. 2024 (SEDIMARK_D5.3), Storage of Provider data assets on multiple stores among other Storage Service Providers. • M36-Sep. 2025 (SEDIMARK_D5.4). Final implementation.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.6 Marketplace services layer

4.3.6.1 Recommendations

Table 28 Recommendations description

Field	Recommendations
Overview	The Recommender component encompasses the Recommendations module and the User Profiling module. The goal is to provide recommendations to SEDIMARK users about assets of interest to them based on their preferences.
Responsible partner	UCD, ATOS
Inputs	The input is the history of user interactions with the SEDIMARK platform, demographic information about the user, and the list of available offerings/assets together with their metadata and offering statistics.
Outputs	The output is a list of recommendations for the user.
Methods used	Content-based recommendation, collaborative filtering, etc.
Datasets used	Any dataset
Language	Python
Modules	Recommendation module, User Profiler

Field	Recommendations
Deployment	As part of the Marketplace Frontend (and backend).
Development timeline	• M13 (Oct. 2023): First prototype. • M15 (Dec. 2023): Improved version. • M18 (Mar. 2024): First Integration with other components. • M27 (Dec. 2024): Second improved version integrated. • M36 (Sep. 2025): Final implementation / tested.
Assumptions, dependencies, and constraints	• Requires logging user interactions, with the SEDIMARK platform (i.e., clicks, searches, purchases). • Requires user profile information (demographic). • Requires metadata about available offerings/assets. • Requires statistics about offerings.
Use case relevance	All
Additional documentation	n/a

4.3.6.2 Offering discovery

Table 29 Catalogue description

Field	Catalogue
Overview	This component is responsible for the provision of a distributed catalogue that holds descriptions of offerings from all Data and Service Providers in a P2P and federated manner.
Responsible partner	SURREY, UC
Inputs	Offering descriptions from Data and Service Providers.
Outputs	Offering descriptions from Data and Service Providers compatible with the Marketplace information model.
Methods used	Formatting, linking

Datasets used	n/a
Language	Python, Java
Modules	RDFLib python library, Apache Jena [31,32]
Deployment	Docker Container
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): First version. • M27-Dec. 2024 (SEDIMARK_D5.3): Second integrated version. • M36-Sep. 2025 (SEDIMARK_D5.4): Final version.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.6.3 Offering sharing

Table 30 Connector description

Field	Connector
Overview	This component is responsible for enabling secured peer-to-peer information exchange between participants. It provides a control plane for contracting, so assets can be securely requested and provisioned in the marketplace. Additionally, it takes part in both the offering registration and sharing flows by hosting the participant self-description and the offering self-listing.
Responsible partner	UC

Field	Connector
Inputs	• Offering descriptions from Data and Service Providers. They might come with a registration request from the corresponding Marketplace supporting tools. • Query to retrieve self-descriptions or particular offerings, coming from authorized components (mainly the catalogue). • Offering negotiation request (from Marketplace supporting tools). • Offering negotiation request (from another participant's connector). • Policy authorization query. • Asset request.
Outputs	• Offering registered in the marketplace. Transaction on the DLT referencing local URL. • Self-descriptions or particular offerings formatted following the marketplace information model. • Triggers an offering negotiation request to another participant's connector. • Triggers the offering negotiation procedure between participants. • Policy authorization response. • Asset retrieval from provider backend.
Methods used	n/a
Datasets used	n/a
Language	Java, Typescript
Modules	Eclipse Data Connector, Non-SQL backend (possibly Apache Jena), ODRL (Open Digital Rights Language) enforcement engine
Deployment	Docker container

Field	Connector
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2), Initial version, basic functionality without security layer integration. • M27-Dec. 2024 (SEDIMARK_D5.3), Intermediate version, with extended functionality. • M36-Sep. 2025 (SEDIMARK_D5.4). Final version, supporting all needed flows.
Assumptions, dependencies, and constraints	It relies on the DLT Enabler, Trust Enabler and IdM.
Use case relevance	All
Additional documentation	n/a

Table 31 Offering sharing description

Field	Offering Sharing
Overview	This component is responsible for supporting the offering description sharing.
Responsible partner	SURREY, UC
Inputs	Offering Request
Outputs	Offering Self-description and Offerings.
Methods used	Formatting, Linking, Indexing
Datasets used	n/a
Language	Python, Java, RESTful API, SPARQL [38]
Modules	RDF-compatible store
Deployment	Docker container

Field	Offering Sharing
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2), Initial integration of Offering Sharing component with Connector API. • M27-Dec. 2024 (SEDIMARK_D5.3), Second integration of Offering Sharing component with Connector API. • M36-Sep. 2025 (SEDIMARK_D5.4). Final integration of Offering Sharing component with Connector API.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.6.4 Offering description

Table 32 Description of Offering description

Field	Offering description
Overview	This component is responsible for the offering generation based on existing assets.
Responsible partner	INRIA, UC, ATOS
Inputs	Participant knowledge about its own assets, annotations from system pipelines.
Outputs	Offering from Data and Service Providers compatible with Marketplace information model.
Methods used	Formatting, linking
Datasets used	n/a
Language	Python, Java, TypeScript
Modules	RDFLib, other parsing libs
Deployment	Docker Container

Field	Offering description
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2), Initial version of offering Information Model defined, manual generation. • M27-Dec. 2024 (SEDIMARK_D5.3), Intermediate version, additional supporting tools. • M36-Sep. 2025 (SEDIMARK_D5.4). Final version, with semi-automatic generation.
Assumptions, dependencies, and constraints	Participants are the ones deciding how to package their own assets for publication, so they are responsible for offering generation
Use case relevance	All
Additional documentation	n/a

Table 33 Marketplace IM validator description

Field	Marketplace IM validator
Overview	This component is responsible for the validation of the Marketplace Information Component, with special emphasis on offerings.
Responsible partner	INRIA, UC
Inputs	Offering from Data and Service Providers.
Outputs	Validated offering following the Marketplace information model.
Methods used	Formatting, linking
Datasets used	n/a
Language	Python, Java, TypeScript
Modules	RDFLib, other parsing libs
Deployment	Docker Container

Field	Marketplace IM validator
Development timeline	• M18-Mar. 2024 (SEDIMARK_D5.2): Initial version. • M27-Dec. 2024 (SEDIMARK_D5.3): Intermediate version. • M36-Sep. 2025 (SEDIMARK_D5.4): Final version.
Assumptions, dependencies, and constraints	n/a
Use case relevance	All
Additional documentation	n/a

4.3.6.5 Frontend

Table 34 Marketplace GUI description

Field	Marketplace GUI
Overview	The marketplace GUI consists of a web frontend enabling users to: • Register new participants and manage their accounts. • Browse the offerings catalogue. • Add or remove new offerings. • Negotiate contracts between offerings providers and consumers. • View statistics about their provided/consumed offerings. • Access data processing orchestrator.
Responsible partner	ATOS
Inputs	The inputs vary largely depending on the purpose of the marketplace usage. They can be grouped into several categories: • Users' credentials for authentication. • Descriptions of participants, offerings and contracts. • Statistics for transaction monitoring or offering recommendation. • Requests to internal SEDIMARK components (logging, data processing, storage, etc.).

Field	Marketplace GUI
Outputs	Outputs also span a wide spectrum depending on marketplace GUI usage. The frontend may display: • Information about the authenticated participant account. • A subset of the offerings in the catalogue depending on the user's role(s) and search queries. • Offering descriptions and statistics. • Past and ongoing transactions. • Currently running data processing pipelines.
Methods used	• Shape the personal characteristics of each user to clearly define needed features. • Design mock-up UIs and prototypes inspired by existing marketplaces. • Audit necessary interactions with other SEDIMARK internal components. • Iteratively improve frontend design based on consortium partners' feedback.
Datasets used	Any datasets used or needed from the list of datasets in section 4.2.
Language	JavaScript/Typescript, React
Modules	Next.js [39], Tailwind CSS [40]
Deployment	Docker container.

Field	Marketplace GUI
Development timeline	• M15 (Dec. 2023): Preliminary version with some basic features (catalogue browsing, offering registration) and additional mock-up UIs to agree on design principles. • M18-Mar. 2024 (SEDIMARK_D5.2): First version covering required functionalities. • M27 (Dec. 2024): Second version validated with partners, featuring a demonstration of transactions involving use cases assets. • M36-Sep. 2025 (SEDIMARK_D5.4): Final version.
Assumptions, dependencies, and constraints	• Users' activity data should remain local and won't be collected in a central system. • Content visible in the front end depends on the user's role. • A non-authenticated user can use the marketplace GUI to browse the catalogue of public offerings.
Use case relevance	All
Additional documentation	• Developer documentation will be written continuously during the implementation. • User documentation, together with some tutorials, will be created after the first version has been validated with the consortium.

5 Integration plan

The primary scope of this chapter is to schedule the timetables for the integration of SEDIMARK modules, define the minimum functionalities for the first version release and incrementally extend the functionalities in the upcoming versions. The plan is based on a “top-down approach". A top-down approach is about breaking down a system into several components that make it up. The process can be repeated to break down components into smaller ones like classes and methods. On the other side, in “the bottom-up approach” development begins with the lowest-level components of the system and progresses upwards towards higher-level components.

Figure 6 below illustrates the system view of the SEDIMARK platform. The system view will be used in Section 5.2 in order to define the step-by-step decomposition for each of the seven scenarios which is the target for the first release.

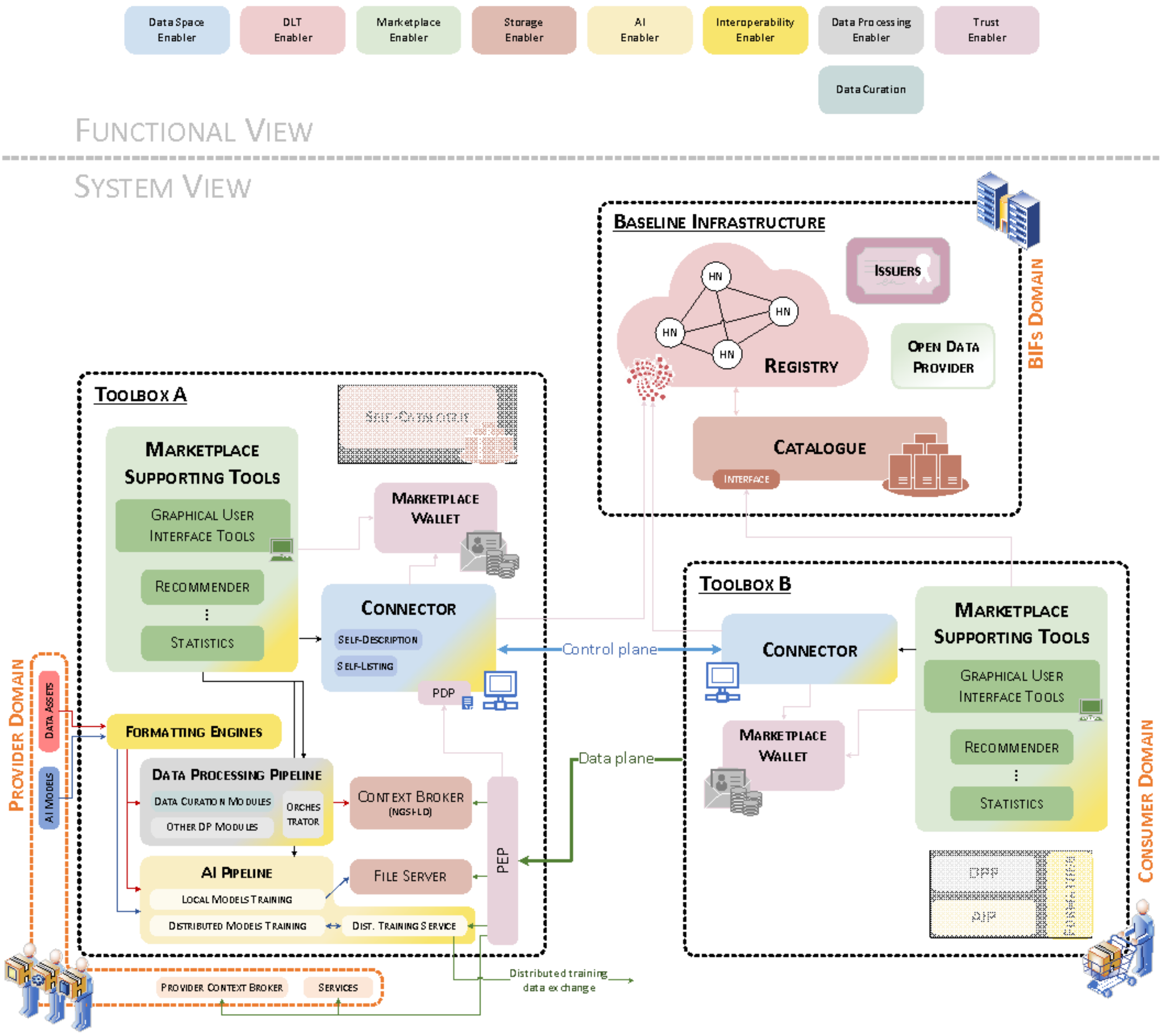

Figure 6 SEDIMARK platform system view

The realization of the integration plan is aligned with the following steps:

Identification: Identify all the modules, building blocks, and interfaces that are primarily involved or depending on the integration with other modules.

Specification: Specify properties and parameters for all the identified modules and interfaces. The modules' assignment to the partner has to be fixed during the whole project. All partners should name the specification needs for each of their modules and each interface involved.

Implementation: The implementation of the modules will follow the specifications determined in the previous step. When implementing the interfaces, the involved partners, being responsible for the modules on both sides of the interfaces, will have to closely collaborate.

Assessment: Evaluate if the implemented interfaces are functional and align with the timetable for integration. The timeline of the integrated SEDIMARK platform is defined in the next section.

Amendment: The results from the integration assessment can be reconsidered, including a redefinition of the modules and probably demand a redesign and a stricter specification of the parameters and limitations set up in the initial steps.

Continuous supervision: The progress of the integration requires continuous monitoring of the interfaces and modules. The partners will conduct monthly consultations (mails, telcos, etc.) to discuss the current progress on development and integration, resolve integration issues and conflicts, and identify new requirements or contingencies.

5.1 Integrated releases of SEDIMARK platform

5.1.1 First version (M18-Mar. 2024)

In the initial phase, the focus will be on implementing the core and minimum functionalities that can provide an MVP (Minimum Viable Product).

The plan for M18 (Mar. 2024) is:

- First version of functional components.
- Supported scenarios PoC.
- Fulfill high-priority requirements (required).
- Making use of data from the 4 pilots.

5.1.2 Second version (M27-Dec. 2024)

At this phase, the plan is to expand the core functionalities and add incremental functionalities identified in the first version.

The plan for M27 (Dec. 2024) is:

- Incremental work and sophistication of components.
- All components are implemented except Payment, Ratings, Tokenization and Open data enabler.
- Fulfill high and medium priority requirements.
- Self-deployable SEDIMARK toolbox with less hard coding.
- Integrated GUI providing access to all functionalities.

5.1.3 Final version (M36-Sep. 2025)

At last, at this stage, the final integration will be ready where all components are in place and the system is optimized for performance purposes.

The plan for M36 (Sep. 2025) is:

- All components implemented.
- First self-deployable SEDIMARK toolbox with no hard coding.
- Fulfil all kinds of requirements (optional, recommended).
- Integrated GUI providing access to all functionalities.

5.2 Supported scenarios and core functionalities for the first version

Considering the three-stepped plan described in Section 5.1, the first iteration of the integration of the SEDIMARK platform is meant to support all the functionalities in charge of fulfilling the high-priority requirements. In this sense, rather than an integrated platform, a set of fundamental scenarios have been defined. These scenarios encompass the key procedures and situations that the SEDIMARK platform will have to support. In order to realize each of these scenarios a subset of the components that are being developed in the project will be integrated, thus showcasing a first compounded version of the platform.

However, the seven fundamental scenarios that have been defined, although tightly bound one to each other, will be tackled independently for this first version of the SEDIMARK platform, assuming that the preconditions of any of the scenarios that will be fulfilled at another are already provided.

The seven scenarios have been chosen in such a way that all the mandatory functionalities of the SEDIMARK platform (as they have been elicited in Deliverable D2.1[4] can be demonstrated while keeping the number of independent scenarios as reduced as possible. For this, we have taken as reference the main procedures or situations that will be held by the participants within the SEDIMARK Marketplace.

In this regard, we have envisaged that the Data Providers will first make use of the SEDIMARK platform to improve and assess the quality of their datasets and/or data streams. The Data quality improvement scenario (described in Section 5.2.1) will handle this procedure. Once the Providers are ready to place their assets at the SEDIMARK Marketplace (after having curated their data in the previous scenario, for example), they will have to create and publish the availability of such an asset. This is what the next scenario, the Offering lifecycle scenario (described in Section 5.2.2) will be showing, together with the corresponding discovery of the SEDIMARK Offerings, that is the step that the interested Consumers will have to take before requesting them through the SEDIMARK Marketplace. The Participants onboarding scenario (described in Section 5.2.3), encompasses the steps necessary for Providers and Consumers to acquire their credentials (in the form of Decentralized Identities and Verifiable Credentials) to interact within the Marketplace. Upon the discovery of the Offering representing the availability of one asset that a Provider is willing to exchange through the SEDIMARK Marketplace (i.e., the resulting condition at the end of the scenario described in Section 5.2.2), the Asset (Data) exchange (described in Section 5.2.4) scenario consists of the actual request and transfer of such an asset (a dataset or data-stream, in this case) between the Provider and the Consumer. In Section 5.2.5, the AI-related scenarios will show several situations employing different AI-based mechanisms that are supported by the SEDIMARK platform and that Providers and/or Consumers will be able to execute, locally or as a service, which would be transacted as an asset, similarly to what the scenario in Section 5.2.4 will be showing. AI-based, intelligent data enrichment is one of the main pillars envisaged for the SEDIMARK Marketplace. Last, but not least, the GUIs and Open Data enabler scenarios (described in Section 5.2.6 and Section 5.2.7 respectively) will present the front-end through which Participants will interact with the SEDIMARK platform (independently of which of the aforementioned scenarios they are involved in) and the capacity of the SEDIMARK platform to leverage data available at Open Data Portals to fulfil Consumers' queries that might not be properly and/or completely addressed by the Providers' Offerings available at the Marketplace.

In the following sections, each scenario's modules, functional entities, step-by-step definition of the scenario, results, and open/missing parts will be described.

5.2.1 Data quality improvement

5.2.1.1 Description

This scenario contributes to the implementation of the data quality improvement functionalities of the SEDIMARK toolbox. It involves the Data Processing Pipeline, the Context Broker, and the File Server functional entities. More specifically, the former will involve all the data curation and quality assessment modules as well as the orchestrator:

- The data adapter to transform the NGSI-LD models into the internal format used by the processing modules.
- The data quality evaluation, profiling, and cleaning modules to assess the data quality of a dataset and its curation.
- The semantic enrichment and data annotation modules to associate some metadata to the datasets, using the results of the data quality assessment.

5.2.1.2 Step-by-step definition

This scenario involves a single provider and is run by a single user. The steps are the following:

- First, we suppose that a dataset exists and is accessible to the user in the Context Broker of the provider.
- The user sets up a processing pipeline involving (at least) data profiling, data quality assessment, and a curation step. He also requires that the result of the latter is stored in the same Context Broker.
- The user requests the processing pipeline to be applied to the dataset.

5.2.1.3 Results

The results of the computations (quality and profiling) are stored in the context broker and accessible to the user for reading. The new dataset is also accessible and can be accessed for further computations.

This scenario can be refined by:

- Including the offering to the choice of the data set.
- Adding the new data set to the offering.
- Run it using the GUI.

5.2.2 Offering lifecycle

5.2.2.1 Description

This scenario contributes to the realization of the establishment and management of the Offering lifecycle, which will enable the registration and publication of Offerings from Providers, and in turn, enable Consumers to search and discover Offerings through a distributed Catalogue. This will be done through a defined set of interactions between modules within the SEDIMARK toolbox and the Baseline Infrastructure. The lifecycle is split into 2 main phases, Registration, and Discovery.

5.2.2.2 Step-by-step definition

For Registration, the scenario involves a single Provider run by a single user and starts after the Formatting Engines have formatted the data according to the Asset Data Model / AI Data Model.

- An Offering description is created through the Marketplace Supporting Tools and Connector which complies with the Marketplace Information Model.
- Prior to publication to the Self-Listing Catalogue, the Offering description is validated by the Validator.
- The Offering is stored locally in the Self-listing Catalogue.
- The Offering endpoint and its corresponding hash are stored in the Registry.

For Discovery, the scenario involves a single Marketplace Operator, a single Consumer and is run by a single user.

- A new instance of a Catalogue is created, which then queries or subscribes to the Registry newly verified Offering registrations and their corresponding reachability information.
- The catalogue retrieves the full Offering descriptions of the new entries via their corresponding Connector.
- The Catalogue indexes the new offerings through centralized or distributed means among Participant nodes. In the first instance, the main Participant node would be the Marketplace Operator utilizing their Baseline infrastructure capabilities.
- The Consumer creates a query request for Offerings through the Marketplace Supporting Tools, i.e., the Web UI or API.
- The Consumer's query is then sent through the Connector to the Catalogue to search for existing offerings.

5.2.2.3 Results

The results of executing the supported scenario are the following:

- Registration
  - The successful formatting, validation of a Provider's offering, and subsequent storage on the self-listing catalogue.
  - The registration of the Offering and the corresponding hash at the Registry.
- Discovery

- The successful retrieval of newly registered Offering (hash) from the Registry to the Catalogue, and subsequent retrieval of the Offering description from the Provider's self-catalogue.
- The successful distribution and creation of the index for the Offerings in the Distributed Catalogue.
- The successful retrieval of Offerings by the Consumer through the search and discovery through the distributed Catalogue.

5.2.3 Participants onboarding

5.2.3.1 Description

- This scenario contributes to the Security and Trust domain within SEDIMARK.
- The implementation of this scenario enables an external user to benefit from the services of the Marketplace.
- The onboarding process takes care of the generation and registration of appropriate digital identities of new users.
- New users will be provided with a new account for the Marketplace, allowing them to interact with the service providers.
- The account will be employed to use different functionalities of the marketplace by the trust layer in conjunction with authentication and authorization policies.

In particular, the onboarding scenario is also valid for internal users of the platform (e.g., a data provider). The digital identities are generated according to the Self-Sovereign Identity (SSI) model.

5.2.3.2 Step-by-step definition

The mandatory step to "use" the Marketplace is to register a new digital identity according to the SSI model. The user needs to:

- Create its own DID (Decentralized Identifier).
- Create its own DID-Document.
- Generate a set of keys (public and private).
- Embed the public keys onto the DID-Document.
- Publish the DID-Document onto the Distributed Ledger.
- Request the Verifiable Credentials (VCs) to the Issuer of the SEDIMARK Marketplace.
- Store the received VCs locally and securely.

It has to be pointed out that the Issuer, before releasing the VC, performs additional steps:

- Retrieves the DID-Document from the distributed ledger.
- Verifies the identity.

In the case of a successful verification, the Issuer:

- Creates the VC.
- Signs the VC with its own Private Key.
- Communicates to the user the VC requested.

If the verification fails, the Issuer does not return any VC to the user.

5.2.3.3 Results

At the completion of this scenario, the user is able to gain a digital identity according to the SSI model for the SEDIMARK Marketplace. The VC obtained is stored locally and securely together with the private key of the identity generated. Additional elements stored are the DID, the DID-Document, and the public key of the user.

5.2.4 Asset (Data) exchange

5.2.4.1 Description

This scenario plays a central role in advancing the capabilities of the data space related functionalities. This concept revolves around creating a marketplace where different assets can be bought, sold, and exchanged among various participants. This bridges the gap between providers and consumers, fosters collaboration, and enables efficient access to a wide range of assets.

The scenario encompasses two complementary planes:

- The control plane ensures the smooth governance of data transactions, with participants able to agree on access controls, pricing models, and licensing terms to protect their interests and maintain data privacy.
- The data plane, on the other hand, underpins the technical infrastructure that enables the exchange of assets. It employs different technologies like secure APIs, data streaming, and encryption to facilitate the seamless flow of information from provider to consumer and to ensure that data is exchanged efficiently, securely, and in compliance with regulatory frameworks.

5.2.4.2 Step-by-step definition

The steps of the scenario are depicted in Figure 7 and listed below:

1) Query offering details (Connector).
2) Contract details are agreed by consumer (Connector).
3) Consumer signs a smart contract (Connector => Registry).
4) Consumer notifies that the smart contract is signed (Connector).
5) Provider signs smart contract (Connector => Registry).
6) Provider notifies that the smart contract is signed (Connector) plus Agreement (reference of smart contract).
7) Provider sets the PDP (Policy Decision Point) and PEP (Policy Enforcement Point) to allow access to consumer (Connector).
8) Consumer requests data (Connector).

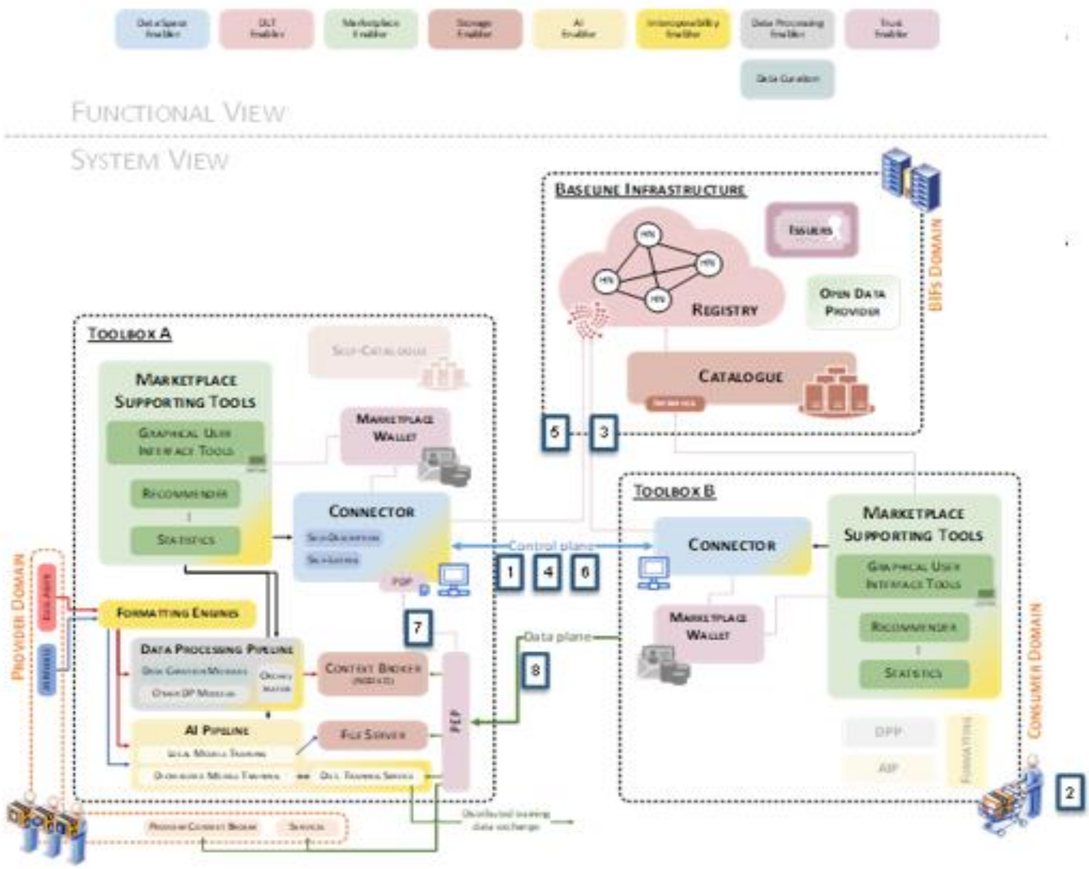

Figure 7 The procedures in the Asset (Data) exchange scenario

5.2.4.3 Results

The results achieved in this scenario touch 3 different domains:

- First, it enables negotiation and agreement on access to a set of assets included in a particular offering.
- Second, the assets available within the provider's domain (e.g., in a Context Broker) are accessible from the Consumer's Toolbox.
- Last, all these operations are carried out in a trustworthy manner, being authenticated through participants' credentials at their respective wallets.

5.2.5 AI-related scenarios

5.2.5.1 Description

The AI pipeline in SEDIMARK is the group of components aiming to provide intelligence in the project, supporting the development of AI models and tools to build knowledge on top of the shared datasets. As discussed in Deliverable SEDIMARK_D2.2, the AI pipeline (or AI Enabler as it is also called) can be split into two main parts that deal with (i) local model training and (ii) distributed model training. The local model training consists of a set of modules that build and run the AI models locally at the participant's premises only with its own data and work as standalone components. The distributed model training aims to leverage the data of multiple participants and jointly build AI models for better generalization. The AI-related scenarios in the trials will cover all the different cases for local and distributed training and provide some insights into how well they perform in a real environment.

5.2.5.2 Step-by-step definition

Local model training

This comprises techniques for model training locally on a SEDIMARK node with local access to training data to enable a parameterized machine learning algorithm to output a model with optimal learned trainable parameters that minimize an objective function. Local models separate the data into local groups and apply a different model for each group.

The steps of the indicated scenario are the following:

- User defines the dataset to build the local model.
- Set up a data quality processing pipeline with all the required pre-processing steps (data profiling, anomaly/outlier/noise detection, duplicate detection, missing value, etc.).
- User initiates the local training process and decides the model applied to the given dataset.
- Forwards the model to the AI orchestrator with the description of the AI / ML mode. Is to be executed.
- Training execution.
- Tuning parameters and hyperparameters and improving model accuracy.
- Select options for optimization.

Service-Shared Distributed Model Training

This scenario deals with the distributed model training in SEDIMARK. We identify two different actors for this scenario:

- The data consumer or user, a person using the SEDIMARK platform.
- The data provider, data source or agent, that is, the computer system that contains the platform tooling such as the SEDIMARK connector.

This scenario describes a set of agents having the capability to train machine learning algorithms and data analytics at the data sources. The user will be able to use the agents connected to the SEDIMARK platform to jointly train machine learning models, more specifically neural networks, without the agents having to share their potentially confidential data with competitors. The focus of this scenario is on synchronous, asynchronous, and decentralized federated learning. More specifically, we deal with the situation when the data consumer is the one starting the use of the federated learning model training services (the other situation, i.e., when the data provider wants to improve the quality of the data, will be handled in the second federated learning scenario described below).

Figure 8 illustrates the involved steps in the described scenario. The steps involved in the scenario are as follows:

- The data consumer purchases the required data and computing capabilities from the data providers inside the SEDIMARK platform.

- The data consumer writes a fleviscript and specifies a model in Keras v3 format that describes the federated learning program. This can be synchronous, asynchronous, or decentralized. However, with fleviscript, the data consumer has much more flexibility to write different kinds of federated learning protocols.
- The data consumer sends the fleviscript and Keras v3 serialized model to the data providers who will handle that information to the flevi-interpreter.
- The flevi-interpreter will spawn a fleviden instance that will handle all the execution of the federated learning protocol, including communication with the other specified data providers.
- At the end of the training process, fleviden will coordinate as specified by the flevi-script, to hand over the resulting global model back to the data consumer.

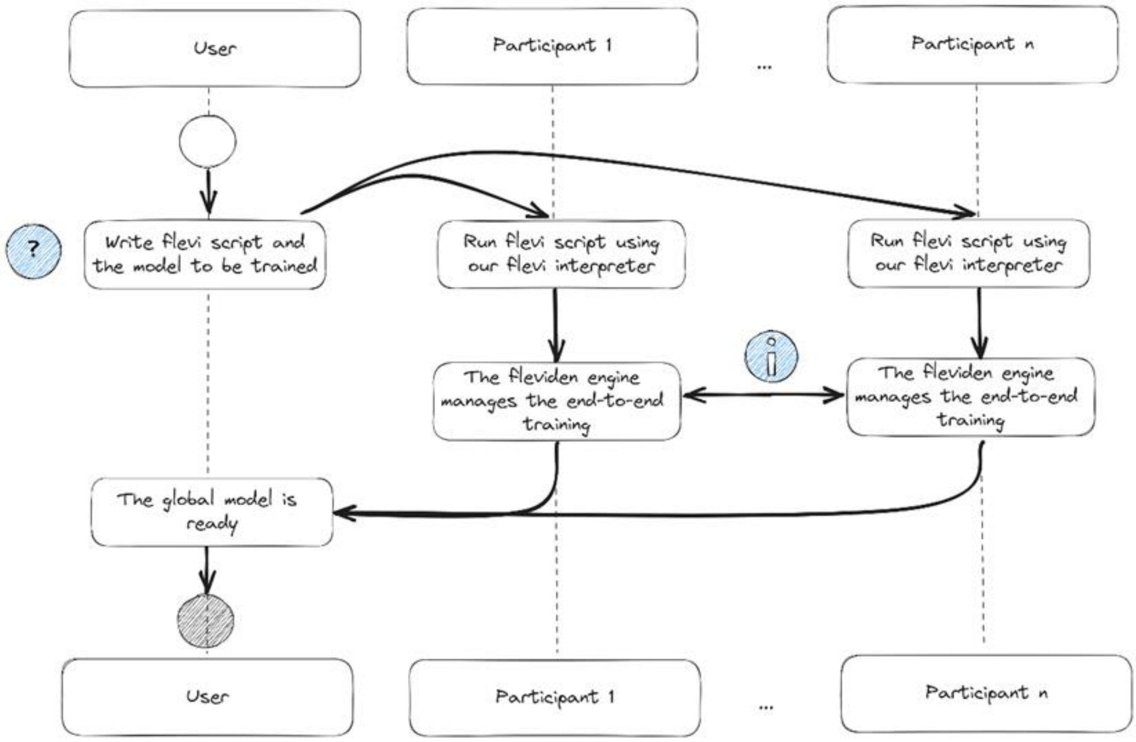

Figure 8 A sequence diagram illustrating the federated learning scenario in which the data consumer or user triggers a new federated learning protocol in SEDIMARK.

Model-shared Distributed Model Training

This scenario provides the provider-initiated process for distributed model training, enabling a continuous and dynamic process for training. The steps of the process are the following:

- Provider has some data and wants to build a model on it.
- Provider initiates the distributed training process and starts building a local model based on model configuration and training parameters.
- Provider's local distributed model training module (shamrock.AI) starts and creates a server listening for connections from the clients.

- Provider creates an Offering for the distributed model as an asset, by selecting an initial version of the model and defining the necessary parameters.
- Provider shares the offering on the SEDIMARK marketplace.
- Other interested participants are searching the marketplace to find models and datasets.
- An interested participant finds the offering for the distributed model and "purchases" it.
- By purchasing the model, the participant (a.k.a. client) downloads the model and the training configuration.
- The Client starts their own distributed training module as a client, loads the model, and connects to the Server (Provider) for exchanging the weights.
- New clients can discover the model and participate in the process and existing clients can leave in a dynamic way.
- The process above is described for Federated learning, with a server-client approach. A similar process can also be described for Gossip Learning, with the main difference being that in Gossip Learning there is no client-server approach, and everyone is a client randomly selecting at each round to which other clients to send the model updates.

5.2.5.3 Results

The scenarios will result in the building of ML/AI models either in a standalone or a cooperative way together with other participants. The models are considered trained when they have converged to some value and further training doesn't improve the result. This can be realized in comparison with some pre-existing model trained centrally on similar data or when test accuracy/loss doesn't change significantly.

5.2.6 GUIs

5.2.6.1 Description

The marketplace GUIs are the entry point for users to interact with most of the SEDIMARK platform components using a set of user-friendly graphical interfaces. Consequently, its usage scenarios consist mostly of ensuring that the scenarios already described in previous sections of 5.2, covering the core functionalities of SEDIMARK, can be actioned via the web frontend. The marketplace UIs also provide additional features to improve user experience such as offering recommendations during catalogue browsing, or access to other SEDIMARK components to build data/AI processing pipelines. To summarize, the following scenarios are covered:

- User authentication and account management.
- New participant registration: for the MVP, this feature may not be implemented. In that case, it will be assumed that participants have been created by administrators, before using the marketplace.
- Offerings catalogue browsing.
- New offering registration.
- Consumed/provided offerings management (past and ongoing transactions).

- Access to data processing orchestrator: not in MVP.
- Access to AI orchestrator: not in MVP.

5.2.6.2 Step-by-step definition

User authentication and account management

This scenario consists of providing a secure authentication mechanism satisfying the following properties:

- Users can log in easily.
- Users can reset their password (not in MVP).
- Non-authenticated users can only access the catalogue of public offerings.
- Authenticated users can see/edit their account data.

New participant registration

This scenario covers:

- The registration of a new organization.
- The registration of new participants within the organization.

The MVP of the SEDIMARK platform won't prioritize this scenario, yet it will be implemented in the first production version of the platform.

Offerings catalogue browsing

The frontend should enable users to:

- Browse the offerings catalogue, seeing only those corresponding to their roles.
- Filter the displayed offerings depending on various search queries' parameters such as creation date, offering name or description.
- Get recommendations on offerings they could be interested in consuming, depending on their activity in the marketplace (not in MVP).

New offering registration

The frontend will contain a page to enable users to:

- Create a new offering from scratch, with some helpers to provide a valid self-description of it. As a first step, this offering will be a dataset or a set of files.
- Validate the offering description.
- Redirect users toward the offerings management page once the valid offering description has been registered.

Offerings management

This page will boost the following features:

- Navigating through the transactions of the user, with sorting and filtering tools (date, name, consumed/provided offerings only).
- Displaying the statuses of the transactions of the user (done, ongoing, etc.).
- Accessing the detailed history of the transactions.
- Providing usage statistics about the offerings (not in MVP).

Access to data processing and AI orchestrators

This scenario illustrates how the data processing and AI orchestrators can be accessed directly via the marketplace GUI, so users can conveniently build and use these tools to shape offerings they would like to provide or refine offerings they consume. The SEDIMARK MVP does not require such access to be in place, but it will be present in future versions.

5.2.6.3 Results

The marketplace GUIs scenarios can be considered achieved once:

- A visiting user (non-authenticated) can browse the catalogue of public offerings in the front end, and filter with search queries.
- A participant in the SEDIMARK ecosystem can authenticate and manage her/his account.
- A participant can browse the catalogue of offerings, and filter results with search queries.
- A participant can see highlighted offerings, recommended based on his/her marketplace browsing activity (not in MVP).
- A participant can add a new offering to the catalogue, the underlying asset being data.
- Two participants can negotiate a contract to exchange the data described in the offering.
- The provider and the consumer can both monitor the transaction in a dedicated dashboard.
- A participant can access the data processing orchestrator and build a pipeline (not in MVP).
- A participant can access SEDIMARK's AI tools for federated learning directly from the marketplace (not in MVP).

5.2.7 Open data enabler

5.2.7.1 Description

The Open data enabler aims at provisioning the SEDIMARK catalogue with offerings facilitating access to existing open data portals directly from the marketplace. These offerings will be public and free of charge, so any participant can contact them in the marketplace without the need for any negotiation steps.

5.2.7.2 Step-by-step definition

The integration of the open data enabler follows these steps:

- Creation of a dedicated SEDIMARK participant: to fulfill the open data enabler role, this participant only needs a minimal set of components to ensure it can provision the catalogue and execute contracts. Additional toolboxes such as data enrichment, processing and AI are not necessary, at least for its first version.
- First version of open data module: this module is a set of components in charge of exposing open data APIs as SEDIMARK service offerings, hosted in the premise of the open data participant defined in step 1. This first version will target a single open data portal.

- Testing the service offering: once step 2 is achieved, an offering giving access to the open data portal should be discoverable in the catalogue, even by non-authenticated users in the marketplace, and be automatically contractable by other participants. We will use another participant to test such features.
- Extending the open data module to another open data portal.
- Writing documentation and tutorials to facilitate the extension of the open data module to other portals.

5.2.7.3 Results

Once the Open data enabler becomes operational as a participant in the SEDIMARK ecosystem, it should provide public offerings for all open data portals it targets.

5.3 Deployment of software components

Integrating software components using Virtual Machines (VMs), Docker containers, and other orchestration tools involves a series of steps described in the following section.

5.3.1 Deployment steps

Phase 1: Initial Setup and Deployment

- This first step should include the setup of a Virtual Private Server or a Virtual Machine on which all the components will be deployed.
- The Operating System should be Linux based and all partners should be given access to these machines.

Step 1: Setup the Virtual Machines

- Task: Determine the OS, CPU, network, memory, and storage requirements for each component. Allocate VMs accordingly.
- Tools: Virtualization software like VMware [24], Hyper-V [25], or cloud based VMs from AWS, Azure [26], etc.
- Security: Establish user accounts and setup access control. Distribute access credentials to partners.
- Additional: Document the chosen configuration for future reference and updates.

Step 2: Installation of Prerequisites

- Task: Configure any OS-level security settings, such as firewall rules, to ensure the components from within the Docker images will be accessible.
- Tools: Package managers apt for Ubuntu [27], yum for CentOS [28]. These prerequisites are to support Docker installation and management.
- Additional: Keep an updated list of all prerequisites for documentation and troubleshooting.

Step 3: Deployment of Docker Images

- Task 1: Copy docker images containing the various components to the VM and run them.
- Task 2: Test the containerized applications to ensure they're running as expected.

- Task 3: Use Docker Compose if there are dependencies between containers or if multi-container setups are needed.
- Tools: Docker CLI, Docker Compose for multi-container apps.
- Additional: Employ CI/CD pipelines to automate the build and deployment of Docker images.

5.3.2 Integration steps

Step 1: Inter-Component Communication

- Ensure that each containerized component can communicate with others. This might involve setting up Docker networking or linking containers.
- Configure any necessary environment variables or configuration files that dictate how the software components interact.
- Perform testing of process workflows that involve several components that are interlinked to identify if any component is not correctly deployed.

Step 2: Set Up Monitoring and Logging

- Install and configure monitoring tools (e.g., Prometheus, Grafana).
- Route logs from the containers to a centralized logging system (e.g., ELK stack).
- Set up alerts for any critical or error-level logs or metrics.

Step 3: Implement Continuous Deployment/Integration

- Set up a CI/CD pipeline to automatically test and deploy updates to the software components.
- Ensure that this pipeline can handle rolling back faulty deployments.
- Document the CI/CD process, including how to push changes and trigger deployments.

5.3.3 Deployment modules

These modules summarize the steps described above as these items are not to be missed in any integration process.

- **API contracts**: All components should have a consensus on API agreements, data structures, and communication standards.
- **Data coordination:** Essential data, if interdependent, should be readily available to the necessary components.
- **Validation**: Utilizing integration validations, the system's resilience against unforeseen scenarios or potential breakdowns is ensured.
- **System insight**: By introducing system monitoring and logging mechanisms, the stability of the integrated system can be gauged effectively.
- **Safeguarding**: Incorporating safety features like API credentials and encrypted connections ensures that interactions between components remain secure.

In the rapidly evolving technological landscape, integrating software components in distributed systems using Virtual Machines (VMs), Docker containers, and orchestration tools is the default status. Such an approach facilitates scalability, fault tolerance, and decentralized management cornerstones of modern software architecture. The steps outlined for deployment and integration emphasize the importance of ensuring smooth inter-component communication, robust monitoring, and an agile approach to deployment and updates.

However, considering that the SEDIMARK platform is a distributed system of data services as well as AI services where transactions are secured by IOTA tangle, the deployment steps can be changed accordingly. These modifications will be captured in the following versions of this document.

6 Definition of evaluation framework and performance metrics (per use case)

6.1 Evaluation methodology

This section presents the methodology that will be used to evaluate the performance of the SEDIMARK system and the various criteria to be used for this evaluation. The criteria include technical criteria specifically crafted for each technique/module used in the evaluation, as well as overall criteria/KPIs specific for each of the use cases.

SEDIMARK, as an innovation action, aims to build a proof of concept set of tools that can be used to instantiate a network of participants and allow them to build a marketplace and start exchanging assets. To perform the evaluation on this set of tools, the tool developers have provided a set of criteria for the key system components that will be the target of the evaluations.

The SEDIMARK evaluation methodology is inspired by two standards defined by ISO regarding evaluating the quality of software products. ISO/IEC 9126 [7] is a series of standards that specify criteria and metrics for product quality in software engineering, as well as a simplified evaluation process. This series of standards was replaced in 2011 by the ISO/IEC 25010:2011 SQuaRE standard [8], which also adds security and compatibility as main characteristics [9]. These standards split the process into four parts including:

- Quality model assessment related to functionality, reliability, usability, efficiency, maintainability and portability.
- Internal metrics.
- External metrics.
- Quality-in-use metrics.

Out of all these, the main focus of SEDIMARK evaluation will be on the "quality model assessment" and especially the "functionality" metrics for individual components and the system as a whole, whereas less emphasis will be given on usability, reliability, etc.

The evaluation methodology is also inspired by ISO/IEC 14598 [10], which provides requirements and recommendations for implementing in practice software product evaluations. This standard was revised in 2011 by the ISO/IEC 25040:2011 [9] standard, which splits the evaluation process into five phases: (i) defining the evaluation requirements, (ii) specifying the evaluation, (iii) designing the evaluation, (iv) executing the evaluation and (v) concluding the evaluation. The main actions for each of the phases are depicted in Figure 9:

- **Phase 1**: Establish the evaluation requirements: this phase establishes the purpose of the evaluation process within SEDIMARK and identifies which modules will be included in the process. The quality model here is based on fulfilling the innovation requirements of SEDIMARK.
- **Phase 2**: Specify the evaluation: this phase defines the quantitative and qualitative metrics that will be used in the evaluation and the criteria for the assessment. The metrics can be generic and agnostic to the use cases, but there should also be use case specific KPIs that show the impact of SEDIMARK on each use case separately.

- **Phase 3**: Design the evaluation: this phase includes the planning of the activities for the evaluation process, specifying also what data will be collected, how they will be processed, etc.
- **Phase 4**: Execute the evaluation: this phase is the main phase where the evaluation takes place, running the various evaluation scenarios, getting the measurements, applying the metrics and the decision criteria for the measures and the evaluation.
- **Phase 5**: concluding the evaluation: in this final phase, the evaluation results are revisited, drafting the final report and providing feedback about the results.

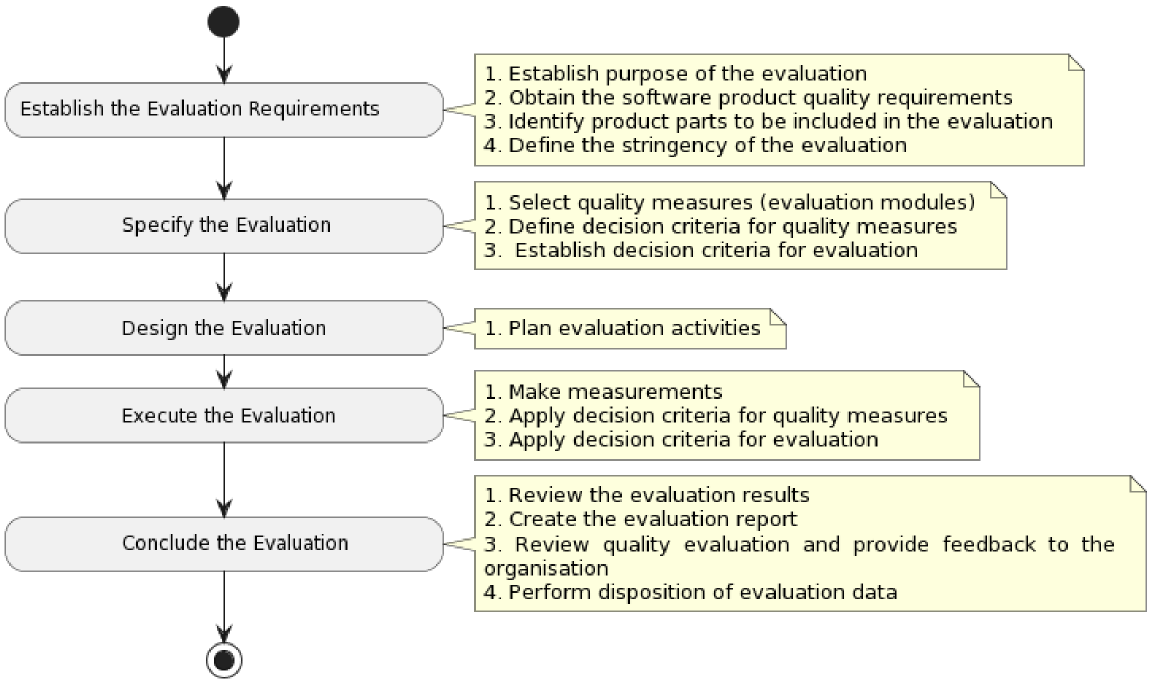

Figure 9 Evaluation process define in ISO/IEC25040:2011 [9]

In this deliverable, the main focus is on the first two phases, including an initial version of the planning of the activities in the third phase. The last two phases will be detailed in the rest of the WP5 deliverables.

6.2 Evaluation process

The SEDIMARK system architecture will be evaluated through its deployment in the four different use case scenarios that were described in D2.1 [4]. The following activities are planned as part of the evaluation process:

- Identification of the requirements and planning of the activities for the evaluation process, identifying the data to be collected, the services to be offered, etc.
- Deployment of the system components for the use case, including
- Deployment of the integrated components (hardware and software).
- Training the involved people who will monitor and participate in the trials.
- Collecting deployment problems as input for the other use cases.

- Execution of the trial scenarios to gather measurements based on the identified KPIs and metrics.
- Evaluation of the measurements through the defined metrics based on the criteria and the KPIs and comparing them against the specified targets.
- Assess first results and provide feedback to the technical work packages to improve the modules, fix problems, etc.
- Perform the second phase of the trials using the updated system modules, executing the above modules again starting from the second bullet.
- Draft the final report for the system evaluation and also provide recommendations and best practices.

6.3 Criteria definition template

In this section, we describe the template for defining the criteria to be used for the evaluation of SEDIMARK. The template is inspired by the RERUM project as presented in [13]. The focus on the evaluation criteria in SEDIMARK is more from the technical perspective and less from the user perspective due to the nature of the SEDIMARK project. The resulting template is depicted in Table 35:

- **ID/name**: a unique I and name for this criterion.
- **Category**: the category of the criterion related to the grouping of the modules into functional enablers.
- **Description**: short text that describes what the criterion is about and why it is considered for the evaluation.
- **Evaluator**: if the evaluation will be done by an "expert" or a simple "user".
- **Evaluation process**: description of how the criterion will be evaluated and on which scenarios.
- **Metrics and targets**: description of the KPIs used for the criterion and what are the target values (it can be Boolean or numeric).
- **Partner**: the partner(s) responsible for the evaluation.
- **Rank**: if the criterion is Mandatory (M), Desirable (D) or Optional (O).
- **Use cases**: the use case(s) where the criterion will be evaluated.

Table 35 Criteria definition template

ID	<unique ID>	Name	<unique name>	Category	<category>
Description	<description of the criterion>				
Evaluator	<user of expert>				
Evaluation process	<how the criterion will be evaluated, i.e., scenarios>				
Metrics	<metrics and target values>				
Partner	<the responsible partners>				

Rank	<Mandatory, desirable, optional>
Use case	<the use cases to assess this criterion>

6.4 Evaluation criteria per module

The next subsections show the evaluation criteria per module for each of the abovementioned scenarios. The assignment for evaluation tasks was proposed by WINGS and UCD.

6.4.1 Criterion table for Data quality improvement

Table 36 Annotation criterion

ID	DataQual.1	Name	Annotation	Category	Quality
Description	Associates some annotations (see 4.3) to a given dataset or AI model.				
Evaluator	Expert				
Evaluation process	This evaluation will be performed at least on one dataset and one AI model. Various annotations are generated and associated: at least one for quality and at least a semantic one. Then, one check that the association is effective in the system.				
Metrics	• The annotations are effectively associated (yes/no). • Time elapsed between the annotation and its availability to other users.				
Partner	EGM				
Rank	Mandatory				
Use case	All				

Table 37 Process criterion

ID	DataQual.2	Name	Process	Category	Quality
Description	Test that a data quality processing pipeline can be set up and run on a given dataset.				
Evaluator	Expert				
Evaluation process	• Follow the process to set up a data processing pipeline with various processing steps. • Request this process to be run on a dataset or a data stream.				

Metrics	• Check that all the processing steps are correctly executed and that the results are stored where expected. • Latency between the processing request and the pipeline is effectively run.
Partner	EGM
Rank	Mandatory
Use case	All

Table 38 Curation criterion

ID	DataQual.3	Name	Curation	Category	Quality
Description	Test the different components for data curation.				
Evaluator	Expert				
Evaluation process	• Prepare a dataset of known quality, with some defects. This dataset can be generated. • Register the dataset into a SEDIMARK use case. • Set up a data quality pipeline with all the required processing steps (evaluation, profiling, cleaning). • Run this pipeline on the dataset.				
Metrics	• Quality evaluation and profiling: accuracy of quality assessments provided, compared to the known quality and characteristics of the generated dataset. • Cleaning: check that the resulting dataset is of the expected good quality by running on it the profiling and quality evaluation steps. • Execution time of each processing step.				
Partner	EGM				
Rank	Mandatory				
Use case	All				

Table 39 Storage criterion

ID	DataQual.4	Name	Storage	Category	Quality
Description	Test the distributed data storage.				
Evaluator	Expert				
Evaluation process	Use internal API to assess some performance metrics.				
Metrics	• Download / upload times as a function of the data size. • Used disk space vs data size. • Size limits.				
Partner	EGM				
Rank	Mandatory				
Use case	All				

6.4.2 Criterion table for Offering lifecycle

Table 40 Offering Registration criterion

ID	OL.01	Name	Offering Registration	Category	Offering Registration
Description	This evaluation will assess the registration of offerings that will be used to populate the Catalogue.				
Evaluator	Expert				
Evaluation process	• Submit Offering to Registry. • Retrieve Offering from Registry to verify successful registration. • Submit Offering in increasing batches for load testing.				
Metrics	• Submission request response time. • Retrieval response time.				
Partner	SURREY, UC, LINKS				
Rank	Mandatory				
Use case	All				

Table 41 Local Catalogue Construction criterion

ID	OL.02	Name	Local Catalogue construction	Category	Local Catalogue
Description	This evaluation will assess the construction of the Local Catalogue.				
Evaluator	Expert				
Evaluation process	Retrieve all Offerings from Registry.				
Metrics	• Compare the number of offerings between all in self-listings and Registry. • Retrieval response time (for probing Connector performance).				
Partner	SURREY, UC, LINKS				
Rank	Mandatory				
Use case	All				

Table 42 Distributed Catalogue Construction criterion

ID	OL.03	Name	Distributed Catalogue Construction	Category	Offering Discovery
Description	This evaluation will assess the construction of the Distributed Catalogue.				
Evaluator	Expert				
Evaluation process	Query Local Catalogues to construct Distributed Catalogues using distributed query mechanisms.				
Metrics	• Compare the number of offerings in all Distributed Catalogue instances and the Registry. • Query response time (for probing Connector performance).				
Partner	SURREY, UC, LINKS				
Rank	Mandatory				
Use case	All				

6.4.3 Criterion table for Participant onboarding

Table 43 Onboarding criterion

ID	ONB.1	Name	Onboarding	Category	Security

Description	Onboarding of a participant within the SEDIMARK domain.
Evaluator	End-User/Expert
Evaluation process	• The participant performs the onboarding process, creating its own DID and the related DID Document. • The participant attaches the DID Document to the Tangle. • The participant requests a VC. • The participant receives a VC. • The participant checks the authenticity and validity of the VC received. • The participant stores the VC securely.
Metrics	The target, from the point of view of the participant that originates the request for a VC, is the correct receipt of the VC from the Issuer.
Partner	LINKS
Rank	Mandatory
Use case	All

Table 44 ID verification criterion

ID	ID.VER.1	Name	ID Verification	Category	Security
Description	Verification of identity.				
Evaluator	End-User/Expert				
Evaluation process	• The participant completes the onboarding process. • The participant requests access to an asset. • The participant sends a VP. • The participant receives the response for access.				
Metrics	The target, from the point of view of the participant that originates the request for assets, is to receive permission from the Verifier. This also verifies authorized access to an asset (access granted/denied).				
Partner	LINKS				
Rank	Mandatory				
Use case	All				

6.4.4 Criterion table for Asset (Data) exchange

Table 45 Successful asset negotiation criterion

ID	ASSET.NEG.01	Name	Successful asset negotiation	Category	Marketplace Service Layer
Description	This evaluation will assess whether the asset negotiation procedure fulfils the required functionality.				
Evaluator	End-user/Expert				
Evaluation process	• A consumer tries to acquire a previously discovered offering. • A contract negotiation procedure is started between the consumer and the offering provider. • After a successful negotiation a contract is signed producing an agreement. • Once the agreement is reached, interactions with the DLT layer are started to tokenize it, thus ensuring its trustworthiness. • A Data Token and/or a VC representing the agreement are exchanged among participants.				
Metrics	• Check that a smart contract including a reference to the mutually agreed information of the agreement has been correctly executed. • Check that the ownership of the Data Token representing the agreement has been properly exchanged. • Check that the relevant (ODRL) policies for accessing the corresponding assets are available as claims on the VC so they can be enforced at the provider's domain PDP.				
Partner	UC, LINKS				
Rank	Mandatory				
Use case	All				

Table 46 Failed asset negotiation criterion

ID	ASSET.NEG.02	Name	Failed asset negotiation	Category	Marketplace Service Layer
Description	This evaluation will assess whether the asset negotiation procedure fulfils the required functionality.				

Evaluator	End-user/Expert
Evaluation process	• A consumer tries to acquire a previously discovered offering. • A contract negotiation procedure is started between the consumer and the offering provider. • The negotiation fails because the consumer is not able to meet the requirements imposed by the provider.
Metrics	The target, from the point of view of the participant that acquires a particular offering, is the denial of access to the asset, expressed by the lack of existence of a related smart contract and the corresponding policies in the provider's domain PDP.
Partner	UC, LINKS
Rank	Mandatory
Use case	All

Table 47 Data asset provisioning criterion

ID	ASSET.PROV.01	Name	Data asset provisioning	Category	Marketplace Service Layer
Description	This evaluation will assess whether, once an agreement is in place, the asset provisioning mechanism allows a consumer to retrieve a data asset described within an offering.				
Evaluator	End-user/Expert				
Evaluation process	• A consumer successfully acquires a previously discovered offering and the corresponding agreement is in place. • The consumer requests access to data assets listed in the offering, including the corresponding VP with the agreed access policies. • Access policies are enforced. • Data asset is retrieved using the NGSI-LD Context Broker API for either datasets or data streams. • The participant receives the data asset, either as a single synchronous response or as several asynchronous notifications.				
Metrics	The target, from the point of view of the participant that originates the request for n assets, is to receive the assets in accordance with the agreed access policies. This also means that if the consumer is not authorized for any reason, he will receive a denial to access the requested assets. Additionally, the criteria can be also evaluated in a distributed storage environment using a context broker federation.				

Partner	UC, LINKS
Rank	Mandatory
Use case	All

Table 48 AI models provisioning criterion

ID	ASSET.PROV.02	Name	AI models provisioning	Category	Marketplace Service Layer
Description	This evaluation will assess whether, once an agreement is in place, the asset provisioning mechanism allows a consumer to retrieve an AI model described within an offering.				
Evaluator	End-user/Expert				
Evaluation process	• A consumer successfully acquires a previously discovered offering and the corresponding agreement is in place. • The consumer requests access to one or more AI models listed in the offering, including the corresponding VP with the agreed access policies. • Access policies are enforced. • AI model is retrieved from the provider's file store (e.g., file system, MinIO). • The participant receives the AI model asset.				
Metrics	The target, from the point of view of the participant that originates the request for n assets, is to receive the assets in accordance with the agreed access policies. This also means that if the consumer is not authorized for any reason, he will receive a denial to access the requested assets. Additionally, the criteria can be also evaluated in a distributed storage environment, using a set of S3 buckets based on MinIO.				
Partner	UC, LINKS				
Rank	Mandatory				
Use case	All				

Table 49 Service provisioning criterion

ID	ASSET.PROV.03	Name	Service provisioning	Category	Marketplace Service Layer
Description	This evaluation will assess whether, once an agreement is in place, the asset provisioning mechanism allows a consumer to consume a particular service described within an offering.				

Evaluator	End-user/Expert
Evaluation process	• A consumer successfully acquires a previously discovered offering and the corresponding agreement is in place. • The consumer requests access to a service listed in the offering, including the corresponding VP with the agreed access policies. • Access policies are enforced. • The request is redirected to the provider's domain where the specific service API is reachable. • The participant receives the service asset.
Metrics	The target, from the point of view of the participant that originates the request for n assets, is to receive the assets in accordance with the agreed access policies. This also means that if the consumer is not authorized for any reason, he will receive a denial to access the requested assets. This target can be initially validated within the toolbox using either the NGSI-LD API supported by the context broker and the S3 API supported by MinIO.
Partner	UC, LINKS
Rank	Mandatory
Use case	All

Table 50 Trust management criterion

ID	ASSET.TRUST.01	Name	Trust management	Category	Marketplace Service Layer
Description	This evaluation will assess whether different offering access policies (possibly described using ODRL) are correctly enforced.				
Evaluator	Expert				
Evaluation process	• Define a set of policies able to cover the envisioned scenarios. • Sequentially, try to access assets protected by all the policies.				
Metrics	Percentage of correctly enforced policies.				
Partner	LINKS, UC				
Rank	Mandatory				
Use case	All				

6.4.5 Criterion table for AI-related scenarios

Table 51 Distributed learning accuracy criterion

ID	DistML.01	Name	Model accuracy	Category	Distributed Machine Learning
Description	This evaluation will assess how accurate is the model trained using the distributed or decentralized training process or if the model loss/accuracy deviates a lot and never converges to a value similar to regular non-decentralized training (which would mean there's an issue in the training process).				
Evaluator	Expert				
Evaluation process	• A distributed learning process will start in one of the scenarios. • Other participants will be part of the process. • After each round, the local models will be evaluated by each participant and the global model will also be evaluated. • Participants will keep the logs locally if it's gossip learning. • At the end of the process, the testing results will be provided to the user along with the trained model, so the user can summarize the results of this criterion.				
Metrics	• Test precision/accuracy variation: if the model's testing precision/accuracy converges to the desired value (the test precision/accuracy of the same model using the same data in a regular, non-decentralized training). • Test loss variation: if the model's testing loss converges to the desired value (the test loss of the same model using the same data in a regular, non-decentralized training). • Training loss: how the training loss varies/drops with training round.				
Partner	WINGS, ATOS, UCD				
Rank	Mandatory				
Use case	All				

Table 52 Distributed learning convergence criterion

ID	DistML.02	Name	Convergence	Category	Distributed Machine Learning
Description	This evaluation will assess how fast the model converges during training.				

Evaluator	Expert
Evaluation process	• A distributed learning process will start in one of the scenarios. • Other participants will be part of the process. • After each round, the local models will be evaluated by each participant and the global model will also be evaluated. • At the end of the training process, the testing results will be provided to the user along with the trained model, so that the user can summarize the results of this criterion.
Metrics	• Speed of convergence: how fast/slow the model converges (i.e., in how many rounds of training) compared with loss of the same model using the same data in a regular, non-decentralized training.
Partner	WINGS, ATOS, UCD
Rank	Mandatory
Use case	All

Table 53 Distributed learning communication cost criterion

ID	DistML.03	Name	Communication cost	Category	Distributed Machine Learning, Energy Efficiency
Description	This evaluation will assess the communication cost during a distributed training process.				
Evaluator	Expert				
Evaluation process	• A distributed learning process will start in one of the scenarios. • Other participants will be part of the process. • At each round, the participants will measure how many packets they send, and how many bytes per packet. • In FL, the server will aggregate all the results of the participants per round and at the end. • In GL, the aggregation will be done offline.				
Metrics	• Number of packets: number of packets exchanged at each round and during the whole training process. • Bytes exchanged: the number of bytes exchanged per round and in total during the whole training process. • Number of Communication rounds: how many rounds of training are required to train the model.				

Partner	WINGS, ATOS, UCD
Rank	Mandatory
Use case	All

Table 54 Recommendation user acceptance criterion

ID	REC.01	Name	User acceptance	Category	Recommendation
Description	This evaluation will assess how users evaluate the recommendation lists they are presented when they make queries to discover new offerings/assets. The goal is to measure user satisfaction with respect to how they like/dislike the recommended offerings/assets.				
Evaluator	User				

Evaluation process	User:
	• Users will navigate the Marketplace GUI making queries and purchases and also rating offerings.
	• Recommendation system will compute user profiles.
	• Recommendation system will provide recommendations based on user queries.
	• Users will evaluate the recommendations:
	  ○ Direct evaluation using like/dislike for each item.
	  ○ Indirect evaluation, by measuring the number of items users click from the recommended ones.				

Metrics	• Click-through-ratio: number of clicks on the recommendations divided by the total number of recommendations. • User satisfaction: number of liked recommendations. • User dissatisfaction: number of disliked recommendations.				
Partner	UCD, ATOS				
Rank	Desirable (may not have enough real users to provide a non-biased evaluation).				
Use case	All				

Table 55 Recommendation accuracy criterion

ID	REC.02	Name	Recommendation accuracy/ precision	Category	Recommendation

Description	This evaluation test will measure the accuracy of the trained recommendation model. It will be run offline in a simulated context in order to be able to test in full its performance on many users, assuming that in the actual trials, the number of users will be too low to be able to properly measure the performance of the system during training.
Evaluator	Expert
Evaluation process	Expert: • The recommendation model will be trained based on simulated users. • After the model is trained, the model will be tested on new users. • Results for the metrics can also be taken when the system is live during the use case trials.
Metrics	• Top-N Precision: number of relevant recommended items at rank *n*. • Diversity: percentage of similar items recommended within user recommended list. • Personalization: similarity of recommendation lists across users.
Partner	UCD
Rank	Mandatory
Use case	All

Table 56 Recommendation latency criterion

ID	REC.03	Name	Recommendation latency	Category	Recommendation
Description	This evaluation test will measure the latency of service recommendations to the users, assuming that a very high latency will contribute to user dissatisfaction not only with the recommender system but also with the overall SEDIMARK marketplace experience.				
Evaluator	Expert				
Evaluation process	• User submits discovery queries on the Marketplace GUI, requesting offerings/assets. • The Recommender system receives the queries and runs the user profiling and recommender model inference to compute a recommendation list for the user. • Recommender system sends the list to the Marketplace GUI for displaying to the user. • Ending time is when the recommendation is being shown to the user.				

Metrics	Recommendation latency: time difference between user submitting a discovery query and user receiving a recommendation on the Marketplace GUI.
Partner	UCD, ATOS
Rank	Mandatory
Use case	All

6.4.6 Criterion table for GUIs

Table 57 Catalogue browsing criterion

ID	MARK.GUI.1	Name	Catalogue browsing	Category	Marketplace Service Layer
Description	This evaluation will assess whether the offerings catalogue can be successfully browsed in the web front end, and that its content is adequately set depending on the user's role(s). It will also evaluate the quality of the catalogue search engine and the clarity of the displayed results.				
Evaluator	End-user				
Evaluation process	Checklist validating each step, assuming the catalogue has been populated with offerings with various access levels: • A visitor (non-authenticated user) can only see and search through public offerings. • Authenticated users can browse and search through all offerings they are allowed to discover given their role(s). • Authenticated users can see highlighted offerings, recommended to them based on their previous activity in the marketplace.				
Metrics	The focus must be set on the user experience. Despite being subjective, the full catalogue browsing experience will be rated, on a scale from 0 to 5, with the possibility to provide written feedback. Special attention will be given to: • How easy and intuitive searching through the catalogue is. • The quality of the information displayed in the resulting list of offerings (clear, synthetic yet exhaustive enough). • How responsive the search engine is.				
Partner	ATOS, UCD, UC				
Rank	Mandatory				
Use case	All				

Table 58 Offering management dashboard criterion

ID	MARK.GUI.2	Name	Offerings management dashboard	Category	Marketplace Service Layer
Description	This evaluation will assess whether participants can monitor their transactions and manage their offerings within a dashboard in the marketplace UI. This dashboard will provide information about the past and current transactions, as well as offerings usage statistics.				
Evaluator	End-user				
Evaluation process	Checklist validating each step, assuming the participant has both past and ongoing transactions, as a provider and as a consumer: • The user can see all her/his provided offerings, even if they have not been used in a transaction yet. • Past and ongoing transactions can be browsed, and their status and some synthetic information for each of them are displayed. • The list of transactions can be filtered and sorted, by creation date, completion date, status or offering. • Selecting a transaction displays more detailed information, as well as usage statistics of the offering.				
Metrics	The focus must be set on the user experience. Despite being subjective, the full dashboard navigation experience will be rated, on a scale from 0 to 5, with the possibility to provide written feedback. Special attention will be given to: • How easy and intuitive the navigation through the transactions and offerings is (usage of sorting and filtering tools). • The quality of the information displayed in the resulting list of transactions and offerings (clear, synthetic yet exhaustive enough). • How clear and relevant the statistics on the offerings are.				
Partner	ATOS, UCD, UC				
Rank	Mandatory				
Use case	All				

6.4.7 Criterion table for Open data enabler

Table 59 Catalogue browsing criterion

ID	OPEN.MOD.01	Name	Open data module	Category	Open data enabler

Description	This evaluation will assess whether the open data enabler provides public offerings for all open data portals it aims at facilitating access to.
Evaluator	End-user
Evaluation process	The evaluation will be focused on the following aspects: • Any user (visitor or registered participant) can discover all the open data enabler offerings in the catalogue. • Authenticated users can access/view the open data portals' policies prior to contracting the offerings. • Authenticated users can contract the offerings without negotiation.
Metrics	Special attention will be given to the user experience within the process of discovering and contracting the open data offerings. Despite being subjective, contracting open data offerings will be rated, on a scale from 0 to 5, with the possibility to provide written feedback. Moreover, we will verify that: • All expected offerings are present in the catalogue, for any participant or visitor. • Contracting the offerings is free of charge. • The open data offerings' contracts' policies respect the usage policies of the open data portals. • The open data portal access works as intended, i.e., the APIs can be accessed, and data transferred.
Partner	ATOS
Rank	Mandatory
Use case	n/a

6.5 Trials definitions and KPIs

In this section, the criteria analysed in 6.4 will be adapted to the four use cases of SEDIMARK. These metrics will be available to all the different users to perform independent monitoring and evaluation of the platform. The template is like the previous one but with some modifications as shown in Table 60 and in the next bullets:

- **ID**: a unique ID and short name for this criterion.
- **Description**: short text that describes what the criterion is about and why it is considered for the evaluation.
- **Rationale**: brief description of the criterion presence.
- **Evaluation responsible:** Partner who is in charge of the evaluation.
- **Evaluator**: if the evaluation is "expert" or "user".

- **Evaluation process**: description on how the criterion will be evaluated by an “expert” or a “user”.
- **Requirements**: Requirements necessary for the evaluation process.
- **Metrics and targets**: description of the KPIs used for the criterion and what are the targets.
- **Rank**: if the criterion is Mandatory (M), Desirable (D) or Optional (O).
- **Type**: the type of evaluation; lab, trial or both.

Table 60 Criterion definition template for the use cases

ID	<unique ID>	Name	<short name>	Category	<category>
Description	<description of the criterion for the evaluation >				
Rationale	<brief description for criterion presence>				
Evaluation responsible	<name of the responsible partner>				
Evaluator	<Expert or User: {U.Cr.1, U.Cr.7, U.Cr.13}>				
Evaluation process	Expert: <how this criterion must be evaluated by experts> User: <how this criterion must be evaluated by users>				
Requirements	<requirements to proceed with the evaluation>				
Metrics and target	<KPI and target>				
Rank	<Mandatory, desirable, optional>				
Type	<Lab, Trial (or both)>				

Additionally, a trial definition procedure will be implemented per trial site consisting of the steps below:

- Purpose of the experiment
- System Deployment
- Data generated
- Services offered
- Models developed
- KPIs
- Experiment scenarios
- Functional components involved or tested
- Experiment risks

6.5.1 Mobility Digital Twin in Helsinki

The digital twin of Helsinki is formed by a network of interoperable systems, exchanging data over standardized APIs (Application Programming Interfaces) [11]. Geospatial data forms the backbone of the digital twin, on which additional topic specific data assets can be overlaid. The data sets involved in the digital twin also include data describing the traffic environment, in particular the road infrastructure. The open data offering of public administration in the Helsinki region is available from the HRI (Helsinki Region Info share) open data portal [12].

The relationship between the urban digital twin (and city SDI) and data marketplace is two-directional:

- City data is offered to the marketplace (both open and closed data cases exist). There are various reasons for this operating model, the city may e.g., lack suitable storage space or publishing tools, and utilizing 3rd party publishing (e.g., marketplaces) provides better visibility to city's data.
- City may also utilize the data marketplace to gather information on the private data offering, and possibly to procure data to improve processes. The data city is interested in data that may be only available via marketplaces, or city may wish to avoid exclusive procurement and wish to procure service (i.e., access to the data) instead of investment (i.e., ownership).

There are two primary use cases:

- Digital Twin uses data FROM data marketplace.
- Digital Twin provides data TO the marketplace.

In both cases, the functional requirements include:

- The data can be free and public, or it can be e.g., restricted, exclusive or commercial. The metadata describing the availability of the data and licenses has to be maintained and may be available from an external API.
- The usage of/access to the data may be agreed outside the marketplace or within the marketplace.
- The data may be hosted either in the marketplace or in an external service.
- When hosted externally, the systems may use the marketplace as a publishing channel, preferably over an API.

6.5.1.1 KPI table

The system aims to provide data owners with a viable alternative to the traditional public procurement model by offering them a feasible option to purchase and access data. Additionally, the system aims to enable data owners to publish and share their data effectively. To measure the success of these objectives, the following key performance indicators (KPIs) have been established:

Table 61 Data from Mobility DT to SEDIMARK criterion

ID	MobDig.01	Name	Data from Mobility DT to SEDIMARK	Category	Data

Description	Data sharing from Mobility Digital Twin to Data Marketplace.
Rationale	Number of datasets shared from the Mobility Digital Twin to the Data Marketplace.
Evaluation responsible	FV with associated system representative.
Evaluator	Expert
Evaluation process	Expert: SEDIMARK expert validation with the FV/The city of Helsinki support. User: A system user associated with knowledge of the Digital Twin concept with data validation support on SEDIMARK.
Requirements	In both cases, the validation requirements include: • The data can be free and public, or it can be e.g., restricted, exclusive or commercial (traffic counter, air quality). • The metadata describing the availability of the data and licenses has to be maintained and may be available from an external API. • The usage of/access to the data may be agreed outside the marketplace or within the marketplace. • The data may be hosted either in the marketplace or in an external service. • When hosted externally, the systems may use the marketplace as a publishing channel, preferably over an API.
Metrics and target	Target: 3 datasets
Rank	Mandatory
Type	Concept

Table 62 Data from Data Marketplace to Mobility DT criterion

ID	MobDig.02	Name	Data from Data Marketplace to Mobility DT	Category	Data
Description	Data utilization from the marketplace.				
Rationale	Number of datasets obtained from the marketplace and applied in joint visualization in the Mobility Digital Twin.				
Evaluation responsible	FV with associated system representative.				

Evaluator	Expert
Evaluation process	Expert: product owner of Digital Twin or a similar role at FV/The city of Helsinki. User: A system user associated with knowledge of the Digital Twin concept and data validation.
Requirements	In both cases, the validation requirements include: • The data can be free and public, or it can be e.g., restricted, exclusive or commercial (traffic counter, air quality). • The metadata describing the availability of the data and licenses has to be maintained and may be available from an external API. • The usage of/access to the data may be agreed outside the marketplace or within the marketplace. • The data may be hosted either in the marketplace or in an external service. • When hosted externally, the systems may use the marketplace as a publishing channel, preferably over an API.
Metrics and target	Target: 1 dataset
Rank	Mandatory
Type	Concept

6.5.1.2 Trial definition

Purpose of the experiment: Helsinki use case wishes to utilize external data sources as part of its digital twin, to enhance the (local) data economy, and to diversify the options for data acquisition and management.

System deployment: The digital twin of Helsinki is formed by a network of interoperable systems, exchanging data over standardized APIs. Geospatial data forms the backbone of the digital twin, on which additional topic specific data assets can be overlayed. The digital twin approach has also been introduced in the field of mobility. Here the digital twin is a means to combine information from different data sources describing the traffic infrastructure and environment, the traffic itself, and related conditions and context. It thus comprises numerous data sources. So far, the digital twin of mobility has been developed on a conceptual level. However, potential data sources belonging to it already exist and are available from Helsinki.

As the Mobility Digital Twin in Helsinki is being developed as a 'system of systems' at a conceptual level, interoperability and machine readability are emphasized concerning the data marketplace. The next phase of development for the Mobility Digital Twin is currently in progress, to collect all traffic volume and sensor data to Helsinki city's own system on a database (Azure) instance. This initiative is underway as part of The City of Helsinki's Smart Transport Program 2030.

Data generated: The data generated are:

- Infrastructure: The Register of Public Areas in the City of Helsinki contains data about the city's "street and green areas," namely street network as polygons, i.e., the area the street, road or a path occupies, with additional administrative information, such as classification and maintenance responsibilities. The registry is available in WFS format (https://kartta.hel.fi/ws/geoserver/avoindata/wfs). The data is also available at https://kartta.hel.fi/.
- Mobility / traffic: The city maintains a number of automated traffic counters (based on induction loops) that provide data over an open API. Induction loops are physical sensors embedded in the road surface that use electromagnetic fields to detect vehicles passing over them. These loops help in collecting data on traffic volume and patterns, allowing for effective traffic management and planning [32].
- Conditions: Helsinki Region Environmental Services HSY maintains a set of air quality measuring stations providing information on air quality in the city, available over an open interface [33]).

The relationship between the data marketplace and the urban digital twin of mobility is envisioned to become two-directional, as it was introduced at the beginning of section 4.1. As the digital twin of mobility is formed as a "system of systems", the significance of interoperability and machine readability is highlighted concerning the data marketplace.

Services offered: For the use case related to the digital twin of urban mobility, the following SEDIMARK services can be applied:

- Query of available data sets over an API, limited by attributes such as location, tag/classification and timestamp – to be used for retrieving an up-to-date list of datasets available for visualization in the digital twin environment.
- Query of dataset metadata over an API to obtain license information, data source query URL etc. – to be used for retrieving an individual dataset for visualization in the DT environment.
- Discovery & query of available data sources and/or individual data source parameters from an existing data catalogue metadata API, such as CKAN – to be used for listing data sets to SEDIMARK from urban spatial data infrastructure utilized in the DT.

Models developed: The data and data models to be generated are described in Section 4.2.1.

KPIs: The system aims to provide data owners with a viable alternative to the traditional public procurement model by offering them a feasible option to purchase and access data. Additionally, the system aims to enable data owners to publish and share their data effectively. To measure the success of these objectives, key performance indicators (KPIs) have been established and defined in the above tables.

Experiment scenarios: This section should describe all actions of the user and the expected system responses for the planned normal execution of the use case.

- Uploading / linking - the data will be made available for the marketplace, either by uploading it to dedicated hosting providing a URL to an external hosting (such as city's open data portal), or by providing a URL to the data stream (for a dynamic data source over API).
- Metadata production by inputting relevant metadata to the marketplace providing a link to existing metadata, e.g., in an external data-catalogue service.

- Metadata and/or data is provisioned over the marketplace, aggregating statistics of its use.
- Data is provided over UI.
- Data is provided over standardized APIs.

Functional Components involved/tested: The key functional components in the use case are:

- Data Types and Accessibility: Data can be public or restricted, with metadata and licenses available through an external API.
- Data Usage and Access: Agreements can be external or within the marketplace.
- Data Hosting Options: Data can reside in the marketplace or on external services.
- Publishing via Marketplace: When hosted externally, systems can use the marketplace as a publishing channel via an API.

Currently, the mobility digital twin in Helsinki is at a conceptual stage, with potential data sources available from Helsinki, including infrastructure, mobility, traffic, and conditions datasets.

Experiment risks: This section should describe any errors that may result during use case execution and how the system will react or respond to those errors).

- Dataset becomes expired, or maintenance is discontinued.
- Dataset should be flagged accordingly.
- Dataset is removed.
- Data should be marked as removed.
- Party responsible for the data is dissolved.
- Metadata should be updated, and data flagged accordingly.
- Changes to the publicity of the data due to data owner's decisions, changes in legislation etc.
- Problems in real-time data streams.
- Define the level of real-time system should be able to manage.
- There needs to be an alternative way to connect to the data stream.
- Problems in agreements or commercial arrangements between parties à Define the level and role of the system in relation to inter-party agreements or commercial arrangements.
- There must be a manual way to provide access to data.
- Privacy breach.
- Ownership and usage rights must be dealt with within the system or a way to interact securely with external management.
- Potentially include consent management / MyData -features for end user.

Services to be offered Use Case may offer city data to external organizations through the SEDIMARK data marketplace and can utilize external data sources to enhance its digital twin of mobility. These services benefit businesses, researchers, city planners, and other stakeholders by providing access to a wider range of data sources.

6.5.2 Urban bike mobility planning in Santander

In recent years, Santander City Council has made a major effort to evolve mobility patterns to more sustainable and green ones. A network of bicycle lanes has been built throughout the city, recognizing the difficulties associated with this type of initiative. Therefore, it is considered essential to obtain as much information as possible on the use of bicycles, their movement patterns and the use of associated infrastructure (bicycle parking, etc.).

The use case to be developed is based on obtaining direct information on both the movements of bicycle users and the structures that support their mobility. The information is collected using a set of sensors installed on the bicycles themselves and on the infrastructures (bicycle lanes, bicycle parking, etc.).

The data from these sensors, together with other available data deemed appropriate, will feed the marketplace. The aim is to provide a global view of bike mobility in the city with a rich variety of data. However, depending on the specific case, the data may be available in other channels. In fact, the city has in place the so-called Santander Smart City Platform (SSCP) that concentrates in a single repository all the information coming from municipal services and provides a series of associated services for an important set of stakeholders, not only municipal services and decision-makers but also for companies and citizens in general. This platform is the core of the whole system.

In relation to the marketplace, the City Council aims to provide a sufficient number of useful datasets to create a critical mass that will encourage other stakeholders outside the City Council to upload their data and following a “snowball” effect, the set of information will be enriched and gain strength.

6.5.2.1 KPI table

Aligned to the use case description and objectives described in the precedent section a set of KPIs are provided in the following tables:

Table 63 Mobility data from SSCP to Marketplace criterion

ID	BikeMob.01	Name	Mobility data from SSCP to Marketplace	Category	Data
Description	Bike mobility data sharing from SSCP to data Marketplace.				
Rationale	The data flow between the municipality bike mobility available data to the data Marketplace should be measured and maximized.				
Evaluation responsible	SDR, UC				
Evaluator	Expert: Municipality SSCP platform; User: None				
Evaluation process	Expert: Number of counting devices (traffic flow and mounted on bike sensors). User: Number of counting devices (traffic flow and mounted on bike sensors).				

Requirements	Platform endpoint availability.
Metrics and target	Number of elements, target: 4
Rank	Mandatory
Type	Both: Lab and trial

Table 64 Mobility data from Pilot to SSCP criterion

ID	BikeMob.02	Name	Mobility data from Pilot to SSCP	Category	Data
Description	Bike mobility data gathered from pilot and fed to SSCP.				
Rationale	The data flow between the municipality bike mobility pilot gathered data to the data SSCP should be measured and maximized.				
Evaluation responsible	SDR, UC				
Evaluator	Expert: Municipality SSCP platform; User: None				
Evaluation process	Expert: Number of elements counting. User: Number of elements counting.				
Requirements	Platform endpoint availability.				
Metrics and target	Number of elements, target: 4				
Rank	Mandatory				
Type	Both: Lab and trial				

Table 65 Actions triggered by the information provided by the pilot criterion

ID	BikeMob.03	Name	Number of informed decisions	Category	Data
Description	Actions triggered by the information provided by the pilot (new bike lanes, punctual changes in urban infrastructure affecting bike mobility.				
Rationale	Data gathered from the pilot should provide information that gives rise to actions by municipal services.				
Evaluation responsible	SDR				
Evaluator	Expert: Municipality project responsible; User: None				

Evaluation process	Expert: Number of actions User: None
Requirements	Pilot executed and data analysed. Sufficient and reliable information from the pilot.
Metrics and target	Number of actions, target: 3
Rank	Mandatory
Type	Trial

6.5.2.2 Trial definition

Purpose of the experiment: The primary objective of this use case revolves around gathering comprehensive data concerning cycling mobility patterns within the city, encompassing various elements such as infrastructure utilization and user feedback. The ultimate aim is to analyse this data to discern the most effective strategies and actions that can be taken to advance the adoption of environmentally friendly modes of transportation. By leveraging the existing and newly generated datasets, our goal is to draw meaningful insights and formulate a strategic roadmap for promoting the widespread adoption of sustainable transportation modes. This involves identifying not only the most effective actions but also the potential areas for improvement and innovation.

Besides, the enrichment of available data through AI-based mechanisms, coupled with the provision of a diverse and substantial set of datasets through the SEDIMARK marketplace, can act as a catalyst for building a critical mass. This, in turn, serves as an incentive for external stakeholders beyond the City Council to generate new applications and/or actively share their data, creating a self-perpetuating 'snowball' effect that continually strengthens and expands the body of information within the city ecosystem.

System deployment: LoRaWAN based IoT devices installed on bikes, LoRaWAN network infrastructure (GWs), SSCP NGSIv2 Orion Context Broker.

Data generated: The data generated are from bikes (timestamp, geolocation, battery, events either user triggered or generated by sensing and processing) and infrastructure (covered bike parking use; location, use, availability, etc., municipal electrical bikes rental service; timestamp, location, etc., bikes counting in specific bike lanes.

Services offered: The services offered are:

- Data Labelling and Fusion with AI: The process of labelling and annotating diverse data sources and their intelligent combination using AI.3.
- Interoperable Data Sharing via SEDIMARK Marketplace: Creating a unified dataset shared through the SEDIMARK marketplace, facilitating the discovery and understanding of mobility patterns.
- Data Management in the SEDIMARK Decentralized Marketplace: Empowering users and providers with control over their data within the SEDIMARK decentralized marketplace.

- Generation of Information and Adapters for Green Mobility: The creation of information and adapters to support and enhance green mobility initiatives within the city.

Models developed: Bike availability prediction, mobility pattern prediction, data augmentation and enrichment, outlier detection.

KPIs: Number of voluntary participants (10), number of new specific measurements provided by the installed IoT devices (>=4), number of different user generated events (>=2), number of different datasets provided in the marketplace (>= 3).

Experiment scenarios: The experiment scenarios are:

- Data Integration and Analysis Effectiveness: Evaluate the efficiency of AI-based data integration methods in combining heterogeneous data sources related to cycling mobility patterns. Measure the insights gained from this integrated data to assess its effectiveness in informing sustainable transportation strategies.
- User Engagement and Feedback Analysis: Conduct experiments to gauge the level of user engagement and the quality of feedback received from cyclists and other transportation users. Analyse the impact of this feedback on decision-making processes and the identification of areas for improvement.
- Infrastructure Utilization Optimization: Explore scenarios to optimize the utilization of cycling infrastructures based on real-time data. Experiment with dynamic infrastructure allocation strategies to enhance cycling mobility patterns and encourage sustainable transportation.
- SEDIMARK Marketplace Ecosystem Growth: Monitor the growth of the SEDIMARK marketplace ecosystem by assessing the rate at which external stakeholders contribute data and develop applications. Investigate the factors that motivate these stakeholders to participate in the ecosystem.
- Strategic Roadmap Formulation: Conduct experiments to formulate a strategic roadmap for promoting sustainable transportation.
- Innovation and Application Development: Encourage external stakeholders to create new applications or services using the enriched data available through the SEDIMARK marketplace. Measure the innovation rate and assess the impact of these applications on sustainable transportation initiatives.

Functional Components involved/ tested: Marketplace enabler, Data Space enabler, Data Processing enabler, Interoperability enabler, Storage enabler.

Experiment risks: Lack of enough devices installed on bikes, insufficient coverage by the LoRaWAN gateways and insufficient number of volunteers.

6.5.3 Valorisation of energy consumption and customer reactions/complaints in Greece

The present use case will analyse consumers' energy behaviour and customer conduct in terms of complaints and churn. Two subcases will be defined:

- Energy consumption prediction & clustering (energy oriented – public data): analysis of sparse energy data to predict energy consumption in residential customers, extracting common energy consumption patterns and clustering different regions based on consumption. Prediction, profile extraction and clustering will use the SEDIMARK

AI and data management tools. The SEDIMARK open data enabler will be used to exploit Open Data for weather. Pre-processing involves cleaning for efficient predictions. Processed data and any metadata will be shared in the SEDIMARK marketplace as raw data and as AI service models open to be used by any interested consumer.

- Customer segmentation & churn prediction (customer oriented - private data): development of two AI prediction models which will analyse customer sales and behaviour at a geospatial level. The AI models will focus on: (i) Predict customer segmentation in different regions via postal code, and (ii) Customer churn in different regions via postal code. Customer segmentation & churn prediction will be used privately by Mytilineos S.A. (BU Protergia) for efficiently and feasibly managing existing business customers as well as gaining a better view of the local market for which they are responsible.

6.5.3.1 KPI table

Table 66 Data sharing and validation criterion

ID	Energy.01	Name	Data sharing and validation	Category	Valorisation of energy consumption and customer reactions/complaints
Description	Data Sharing to the Marketplace & Model Validation.				
Rationale	Data accuracy and algorithm evaluation.				
Evaluation responsible	MYT				
Evaluator	Expert				

Evaluation process	Expert: • Prediction Accuracy KPI: Mean Absolute Percentage Error (MAPE) between the model's predictions and the actual values for electrical energy consumption (<5%). • Clustering Quality: Silhouette Score in combination with Davies-Bouldin Index - Evaluate the quality of clustering by calculating an average silhouette score, which measures how well-separated the clusters are and whether data points belong to the correct clusters. Then, measure the average similarity between each cluster and its most similar cluster respectively. • Target: Aim for high silhouette scores and low Davies-Bouldin Index values, indicating well-separated and distinct clusters. User: • Prediction Accuracy: Check the accuracy of the energy consumption predictions for the users' own energy usage. Are the predicted values reasonably close to their actual consumption? Higher prediction accuracy indicates a more successful model. • Clustering Relevance: Users apply their intuition to evaluate the clustering results to see if they make sense for their energy consumption behaviour. Do the clusters align with different usage patterns or user segments that they can identify with? Relevant and meaningful clusters indicate a successful clustering process. • Real-world Impact: Reflect on whether the energy-saving strategies suggested by the system are practical and effective in real-life scenarios. Has the system helped the user reduce energy consumption and lower utility bills? • Adaptability: Evaluate how well the system adapts to changes in the users' energy consumption behaviour over time. A successful process should continue to provide relevant predictions and recommendations as their habits evolve.
Requirements	Data cleaning and anonymisation.
Metrics and target	MAPE <5%, high silhouette score & Low Davies - Bouldin Index.
Rank	Mandatory
Type	Trial (most likely)

6.5.3.2 Trial definition

Purpose of the experiment: Assess the accuracy of the energy consumption predictions made by the AI model and determine how closely the model's predictions align with the actual energy usage of users. Evaluate the clustering process to determine if the AI model successfully groups users with similar energy consumption patterns into distinct clusters.

Assess the accuracy of the AI model's energy consumption predictions on a test data set containing historical energy usage data for a diverse group of users.

System deployment: MYT servers / 2 Models developed as mentioned in the proposal.

Data generated: Energy-oriented data for the energy consumption prediction and clustering will be public and anonymized except for the ZIP Codes. This means that the only data that will have public status are the weather data, the residential size and any residential consumption related data. Customer-oriented data regarding segmentation and churn prediction will be private and anonymised.

Services offered: Query of available / public source data and Query of available / public data that are a result of the ML algorithms.

Models developed: Decision tree, supervised learning, K-means, unsupervised learning

KPIs: The main KPIs are:

- Objective: Measure the model's ability to make precise predictions and quantify the prediction error using MAPE metric.
- Clustering Performance Scenario: Apply the AI model to cluster users based on their predicted energy consumption patterns.
- Objective: Evaluate the quality of clustering results using metrics such as silhouette score and Davies-Bouldin Index. Examine the distinctness and relevance of the identified user segments.

Experiment scenarios: Provision of data through API or direct upload / url of csv file

Functional components: The functional components involved are:

- Customer segmentation and churn prediction
- Energy consumption prediction
- Data analytics

Experiment risks: Data privacy and security risks, bias in data and model, overfitting and underfitting, deployment challenges

6.5.4 Valuation and commercialization of water data in France

In the context of climate change, water is a critical resource that must be managed very carefully. The ecosystem of water management involves many different actors, each having a different responsibility and their own datasets which may be of value for other stakeholders. Currently, these datasets are not or are poorly shared. Allowing different actors to use data of others in an interoperable way may stimulate innovation in water management by allowing new public services and better political decisions.

6.5.4.1 KPI table

Table 67 Number of data providers criterion

ID	Water.01	Name	Number of data providers	Category	Water Valorisation Use case

Description	This evaluation will assess the number of data providers in the water valorisation use case.
Rationale	n/a
Evaluation responsible	EGM
Evaluator	Expert
Evaluation process	Count the number of individual users that provided data to the water valorisation platform through the SEDIMARK platform.
Requirements	n/a
Metrics and target	Count
Rank	Mandatory
Type	n/a

Table 68 Number of data consumers criterion

ID	Water.02	Name	Number of data consumers	Category	Water Valorisation Use case
Description	This evaluation will assess the number of data consumers in the water valorisation use case.				
Rationale	n/a				
Evaluation responsible	EGM				
Evaluator	Expert				
Evaluation process	Count the number of individual users that consumed data from the water valorisation platform through the SEDIMARK platform.				
Requirements	n/a				
Metrics and target	Count				
Rank	Mandatory				
Type	n/a				

Table 69 Number of AI Algorithms deployed criterion

ID	Water.03	Name	Number of AI Algorithms deployed	Category	Water Valorisation Use case

Description	This evaluation will assess the number of AI / Algorithms deployed in the water valorisation use case.
Rationale	n/a
Evaluation responsible	EGM
Evaluator	Expert
Evaluation process	Count the number of AI or other type of algorithms deployed in the water valorisation platform through the SEDIMARK platform.
Requirements	n/a
Metrics and target	Count
Rank	Mandatory
Type	n/a

Table 70 Number of datasets in the catalogue criterion

ID	Water.04	Name	Number of datasets in the catalogue	Category	Water Valorisation Use case
Description	This evaluation will assess the number of datasets in the catalogue in the water valorisation use case				
Rationale	n/a				
Evaluation responsible	EGM				
Evaluator	Expert				
Evaluation process	Count the number of datasets in the catalogue of the water valorisation platform				
Requirements	n/a				
Metrics and target	Count				
Rank	Mandatory				
Type	n/a				

Table 71 Number of far edge devices deployed criterion

ID	Water.05	Name	Number of far edge devices deployed	Category	Water Valorisation Use case
Description	This evaluation will assess the number of far edge devices deployed in the water valorisation use case.				
Rationale	n/a				
Evaluation responsible	EGM				
Evaluator	Expert				
Evaluation process	Count the number of far edge devices that use SEDIMARK components deployed in the water valorisation platform.				
Requirements	n/a				
Metrics and target	Count				
Rank	Mandatory				
Type	n/a				

Table 72 Number of open datasets integrated criterion

ID	Water.06	Name	Number of open datasets integrated	Category	Water Valorisation Use case
Description	This evaluation will assess the number of open datasets integrated into the water valorisation use case				
Rationale	n/a				
Evaluation responsible	EGM				
Evaluator	Expert				
Evaluation process	Count the number of open datasets integrated into the water valorisation platform				
Requirements	n/a				
Metrics and target	Count				

Rank	Mandatory
Type	n/a

6.5.4.2 Trial definition

Purpose of the experiment: This use case will have a special focus on the specific aspects of data quality to ensure that they are trusted, to encourage their reuse, and to provide value and new services. For this, the data quality services developed in SEDIMARK will be deployed and experimented in this use case, like validation, curation, and evaluation systems.

System deployment: The system deployed for this an end-to-end FIWARE environments using several of the FIWARE enablers such as DRACO for data ingestion, STELLIO as high-performance context brokering, Superset and Grafana for data visualization. Moreover, an additional web-platform designed for the use-case will be developed and deployed, to add features such as threshold detection and alerting capabilities, data provisioning, data validation and annotation, data export.

Data generated: The main data generated are:

- GIS data.
- Directory of the territory (key persons and institutions).
- Water management and distribution infrastructure.
- Infrastructure monitoring data (pressure).
- Water consumptions (anonymised).
- Weather observation from an external provider.
- Weather observation from station.
- Stocks in water reservoir.
- River, stream flows.
- Piezometry measurements.
- Irrigation programs.
- Soil sensor measurements.
- AI and prediction models result.

Services offered: The main services offered are:

- Tools to integrate heterogeneous data sources.
- Data validation, semantic enrichment, and transformation services.
- Security and authorization policies.
- Edge services for data quality and alerting systems.
- Maps, dashboards, and alerts.
- Basic data marketplace services.

Models developed: The models developed are:

- River water level prediction.
- River flow prediction.

- Water resource availability prediction.
- Irrigation optimisation function.

KPIs: The main KPIs are:

- Number of data providers.
- Number of data consumers.
- Number of AI/Algorithms deployed.
- Number of datasets in the catalogue.
- Number of far edge devices deployed.
- Number of open data sets integrated.

Experiment scenarios: A typical experiment for the water use-case will follow a workflow of this type:

- User uses the data adapter component to convert data from an entity on the context broker to the SEDIMARK internal format.
- The data is processed by at least one of the components listed below.
- The processed data is used as input for a machine learning model to create a new dataset.
- The processed data and the new dataset are converted back to NGSI-LD by the data adapter component.

Functional Components involved/tested: The functional components either involved or tested are:

- Data adapter
- Data quality evaluation
- Error/outlier detection
- Data augmentation
- Missing value imputation
- Data anonymisation
- Data validation
- Data annotation
- Data analytics
- Model inference

Experiment risks: The main experiment risk is that converting data from internal format back to NGSI-LD can be hard especially if different metadata are to be associated with different instances of an attribute (e.g., some data annotations that would be different for each timestamp).

7 Conclusions

This deliverable is the first document of T5.1 “Integration and evaluation plan and methodology” and provides a quite detailed evaluation methodology, performance metrics and integration plan followed in the releases of the SEDIMARK platform.

The integration plan is based on the proposed architectural description and must evolve along with the platform architecture description in the development phases. The approach and overall plan will, however, remain the same. The overall integration strategy has been discussed, and emphasis has been put on the continuous integration/continuous delivery model used, chosen tools and solutions for source code management, code quality, build server and artefact repository. To speed up development, the objective is to implement common functions (e.g., runtime, communication stacks) through docker-based integration layers and develop pipelines to facilitate the integration and validation.

The SEDIMARK platform is divided into three phases / versions: The first version (delivered in M18-Mar. 2024), the second version (delivered in M27-Dec. 2024), final version (delivered in M36-Sep. 2025). The initial release will involve some independent generic scenarios based on the minimum functional components required to support high priority types of requirements. The supported scenarios will follow a consistent format (description, step-by-step definition and results). Furthermore, all scenarios will be tested using data from the project's four pilots. Upcoming versions will include incremental work and component sophistication, as well as increased support for the remaining requirements and less or no hard coding. All functionalities will be accessible via the integrated GUI.

The methodology for evaluating the performance of the SEDIMARK system is also presented. The various evaluation criteria focus on technical aspects specific to each component or technique used in the evaluation. There are also KPI-specific criteria implemented in each use case. The SEDIMARK evaluation methodology is based on two ISO standards (ISO/IEC 9126 and ISO/IEC 14598) that address a product's software quality. The ISO/IEC 9126 standard describes a simplified evaluation process and metrics for product quality, whereas the ISO/IEC 14598 standard specifies requirements and recommendations for implementing software product evaluations in practice.

To summarize, the deliverable is the skeleton of WP5, and its content is critical to the project's future planning. It is linked to previous work and will feed the work package's upcoming tasks: platform continuous integration, use case execution, and platform validation. Aside from that, it is a stepping stone for WP5's next deliverables regarding the versions of the SEDIMARK platform's integrated releases.

8 Bibliography

[1] SEDIMARK, Deliverable 2.2: SEDIMARK Architecture and Interfaces. First version, SEDIMARK, September 2023.

[2] Breuel Cristiano, ML Ops: Machine Learning as an Engineering Discipline., Towards Data Science, 6 July 2021.

[3] Talagala Nisha, Why MLOps (and not just ML) is your Business' New Competitive Frontier., AITrends, 30 January 2018.

[4] SEDIMARK, Deliverable 2.1: Use cases definition and initial requirement analysis, SEDIMARK, Juner 2023.

[5] Open meteo, [Online]. Available: https://open-meteo.com/

[6] Simplifying access to water data, [Online]. Available: https://hubeau.eaufrance.fr/

[7] ISO, "ISO/IEC 9126-1:2001 Software engineering-Product quality-Part 1: Quality model," [Online]. Available: https://www.iso.org/standard/22749.html.

[8] ISO, "ISO/IEC 25010:2011 Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)- System and software quality models," [Online]. Available: https://www.iso.org/standard/35733.html.

[9] ISO, " ISO/IEC 25040:2011 Systems and software engineering- Systems and software Quality Requirements and Evaluation (SQuaRE)- Evaluation process," [Online]. Available: https://www.iso.org/standard/35765.html.

[10] ISO, " ISO/IEC 14598-1:1999 Information technology-Software product evaluation-Part 1: General overview," [Online]. Available: https://www.iso.org/standard/24905.html.

[11] Decision making-Maps and geospatial data, [Online]. Available: https://www.hel.fi/en/decision-making/information-on-helsinki/maps-and-geospatial-data/helsinki-3d.

[12] Helsinki-Open data service, [Online]. Available: https://hri.fi/en_gb/

[13] RERUM, [Online]. Available: https://ict-rerum.eu/

[14] GitHub Actions, [Online]. Available: https://docs.github.com/en/actions

[15] Jenkins, [Online]. Available: https://www.jenkins.io/doc/

[16] Self-hosted runners, [Online]. Available: https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/about-self-hosted-runners

[17] Container registry, [Online]. Available: https://www.redhat.com/en/topics/cloud-native-apps/what-is-a-container-registry

[18] MLOps Venn diagram, [Online]. Available: https://en.wikipedia.org/wiki/MLOps#/media/File:ML_Ops_Venn_Diagram.svg

[19] CI/CD explanation, [Online]. Available: https://resources.github.com/ci-cd/

[20] SEDIMARK, Grant Agreement

[21] Comet, [Online]. Available: https://www.comet.com/site/

[22] Neptune, [Online]. Available: https://aws.amazon.com/neptune/machine-learning/

[23] Weights and Biases, [Online]. Available: https://wandb.ai/site

[24] VMWare, [Online]. Available: https://www.vmware.com/

[25] Hyper-V, [Online]. Available: https://en.wikipedia.org/wiki/Hyper-V

[26] Azure, [Online]. Available: https://es.wikipedia.org/wiki/Microsoft_Azure

[27] Apt for Ubuntu, [Online]. Available: https://ubuntu.com/server/docs/package-management

[28] Yum for CentOS, [Online]. Available: https://en.wikipedia.org/wiki/Yum_(software)

[29] ngsildclient, [Online]. Available: https://pypi.org/project/ngsildclient/

[30] PyLD, [Online]. Available: https://pypi.org/project/PyLD/

[31] rdflib-jsonld, [Online]. Available: https://pypi.org/project/rdflib-jsonld/

[32] Apache Jena, [Online]. Available: https://jena.apache.org/

[33] DAG, [Online]. Available: https://en.wikipedia.org/wiki/Directed_acyclic_graph

[34] LightGBM, XGBoost, Catboost, [Online]. Available: https://neptune.ai/blog/when-to-choose-catboost-over-xgboost-or-lightgbm

[35] GluonTS, [Online]. Available: https://ts.gluon.ai/stable/

[36] BentoML, [Online]. Available: https://pypi.org/project/bentoml/

[37] Rust vs Go, [Online]. Available: https://bitfieldconsulting.com/golang/rust-vs-go

[38] SPARQL, [Online]. Available: https://en.wikipedia.org/wiki/SPARQL

[39] Nextjs, [Online]. Available: https://nextjs.org/learn-pages-router/foundations/about-nextjs

[40] TailwindCSS, [Online]. Available: https://tailwindcss.com/

Annexes

The following tables summarize all the datasets used per trial site and described in Chapter 4.

Table 73 Dataset for Mobility Digital Twin in Helsinki (1)

Field	Information
Type of data	Real time air quality index.
Format of the data	WMS / WFS API, point features with index attribute.
Data model used	Regarding the air quality, no specific data model. Regarding the GIS: JHS 158 (Inspire), https://www.suomidigi.fi/ohjeet-ja-tuki/jhs-suositukset/jhs-158-paikkatietoaineistojen-ja-palveluiden-metatiedot
Data snapshot	Previewing and downloading available at https://kartta.hsy.fi/?zoomLevel=3&coord=25493189.70682079_6678142.5993235&mapLayers=27+100+rasteri,190+100+ilmanlaatu_nyt&uuid=508752a1-2d1e-4011-a0f7-a96e857fff64&noSavedState=true&showIntro=false
Structure/fields of the data	Measuring station name, time, air quality index, measuring station address, measuring station number (geo coordinates).
Size	Very small, megabytes.
Static or real time data	Real time
Data cleaning tasks	No sensitive information, no information on how data is cleaned.
Will there be labelled training data available? If so, how big is the labelled dataset?	No
ML analytical tasks that will likely be performed after the data cleaning task	No

Field	Information
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	Not essential related to this data.
Type of metadata to characterize the data	n/a
Other information	License is CC 4.0

Table 74 Dataset for Mobility Digital Twin in Helsinki (2)

Field	Information
Type of data	Automatic traffic counters (LAM) in Helsinki region.
Format of the data	JSON through API
Data model used	See API descriptions: https://lamapi.azurewebsites.net/swagger/index.html?url=/swagger/v1/swagger.json#/
Data snapshot	See API for previews: https://lamapi.azurewebsites.net/swagger/index.html?url=/swagger/v1/swagger.json#/
Structure/fields of the data	Locations of detectors, traffic volume in last hour, traffic volume in last day, traffic volume in last month, total traffic volume during the time period.
Size	Volume in last day is about 3.5Mb
Static or real time data	Static
Data cleaning tasks	Not knowing how data is cleaned.
Will there be labelled training data available? If so, how big is the labelled dataset?	No
ML analytical tasks that will likely be performed after the data cleaning task	No
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	No edge devices needed.

Field	Information
Type of metadata to characterize the data	Measuring point ID, name, longitude, latitude, vehicle count and speed in 5 minutes (split by driving directions in each measuring point).
Other information	API: https://lamapi.azurewebsites.net/swagger/index.html?url=/swagger/v1/swagger.json#/

Table 75 Dataset for Mobility Digital Twin in Helsinki (3)

Field	Information
Type of data	The Register of Public Areas in the City of Helsinki.
Format of the data	GIS data: polygons through WFS API (XML).
Data model used	Custom
Data snapshot	Not readily, have to grab through WFS API. Data preview at https://kartta.hel.fi/link/cGnmPA
Structure/fields of the data	Key/value pairs, example of few data types: • Street Sections, areas. • Green Sections, areas. • Area maintenance responsible. • Address. • Surface material. • Maintenance rating. • Area index. • Data owner. • Winter maintenance.
Size	Less than 200 Mb.
Static or real time data	Static, updated annually.
Data cleaning tasks	Not done, updating etc. is all manual.
Will there be labelled training data available? If so, how big is the labelled dataset?	No
ML analytical tasks that will likely be performed after the data cleaning task	No

Field	Information
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	No
Type of metadata to characterize the data	https://kartta.hel.fi/paikkatietohakemisto/pth/?id=29
Other information	License CC 4.0.

Table 76 Dataset for urban bike mobility planning in Santander

Field	Information
Type of data	Bikes' positions, bikes' speed, event positions.
Format of the data	JSON
Data model used	Smart Data Model
Data snapshot	dataset_SDR_example.jsonld (@Owncloud 04_WP2\Task2.1\Sample datasets).
Structure/fields of the data	Latitude and longitude coordinates, numerical data for speed, numerical data for tracker battery level, strings for events' type.
Size	Still to be defined, but the initial estimation is: 1 observation per minute (while the bike is moving). Counting with 15 trackers, the rough estimation is around 2000 daily observations.
Static or real time data	Real time data, Offline datasets created ad-hoc.
Data cleaning tasks	Data cleaning and data interpolation are to be done at the Data Processing Pipeline.
Will there be labelled training data available? If so, how big is the labelled dataset?	No
ML analytical tasks that will likely be performed after the data cleaning task	Time series forecasting, deduplication.
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	Data servers.
Type of metadata to characterize the data	Valid ranges for each parameter.

Field	Information
Other information	n/a

Table 77 Dataset for Valorisation of energy consumption and customer reaction/complaints in Greece

Field	Information
Type of data	Energy consumption measurements.
Format of the data	csv
Data model used	AI Based
Data snapshot	n/a
Structure/fields of the data	Numerical data, locations (ZIP codes), text (String), weather data.
Size	5000 unique customers * 8 columns
Static or real time data	Static
Data cleaning tasks	Energy-oriented data for the energy consumption prediction and clustering will be public and anonymized except for the ZIP Codes. This means that the only data that will have public status are the weather data, the residential size and any residential consumption related data. Customer-oriented data regarding segmentation and churn prediction will be private and anonymized. All data processed will concern a specific time range.
Will there be labelled training data available? If so, how big is the labelled dataset?	Energy consumption prediction & churn ==> Yes, we have labelled data, approx. 25% of the dataset. Customer segmentation and clustering ==> No, we do not have labelled data.
ML analytical tasks that will likely be performed after the data cleaning task	Training, testing, validation
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	Our Data Servers.
Type of metadata to characterize the data	n/a
Other information	n/a

Table 78 Valuation and commercialization of water data in France

Field	Information
Type of data	Weather forecasts, water height measurement, flow measurement.
Format of the data	JSON
Data model used	NGSI-LD
Data snapshot	https://stellio-dev.eglobalmark.com/dashboard/dashboard/snapshot/qzpNYnciG6r6ASx1bp5P6sfHvqvrwRCE https://stellio-dev.eglobalmark.com/dashboard/dashboard/snapshot/zf8PmTOcOGYqkSCgK3y3N6MHiI8MKA3v
Structure/fields of the data	Numerical data for weather forecasts plus water measurements, locations (coordinates), strings (name, ID, type).
Size	500 daily values * approx. 20 parameters for weather data.
Static or real time data	Real time data
Data cleaning tasks	We expect to run outlier detection and duplicate detection as data cleaning tasks, and probably interpolation of missing data for more general data processing.
Will there be labelled training data available? If so, how big is the labelled dataset?	No
ML analytical tasks that will likely be performed after the data cleaning task	Times series forecasting.
Do you want to run SEDIMARK tools on extreme edge devices (sensors/gateways) or just on your data servers?	On edge devices.
Type of metadata to characterize the data	Height or depth of measurement for some meteorological parameters. Valid ranges for each parameter can be added manually if needed.
Other information	n/a