Reliable AI and Data Optimization

D2.3 - Design of the General RAIDO Platform and Architecture

Submission date: 31/12/2024

Project Number:	101135800
Project Acronym:	RAIDO
Project Title:	Reliable AI and Data Optimization
Start date:	January 1st, 2024 **Duration:** 36 months
Deliverable:	D2.3 - Design of the General RAIDO Platform and Architecture
Work Package:	WP2
Lead partner:	UBITECH-LIM (UBI)
Author(s):	Christos Kotronis (UBI), Giorgos Theodorou (UBI), Ioanna Ntinou (QMUL), Georgios Tzimiropoulos (QMUL), Nikos Androulidakis (8BELLS), Vasileios Lolis (CERTH), Yannis Spyridis (KU), Georgios Fragkoulidis (AXON), Andreas Foteas (AXON), Ilias Siniosoglou (MINDS), Georgios Tsoumplekas (MINDS), Georgios Tziolas (SID)
Reviewers:	Ioanna Ntinou (QMUL), Georgios Tzimiropoulos (QMUL)
Due date:	31/12/2024
Deliverable Type:	R—Document, report **Dissemination Level:** PU—Public
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	01/11/2024	Christos Kotronis (UBI)	First release of Table of Contents (ToC).
0.2	03/11/2024	Christos Kotronis (UBI), Georgios Theodorou (UBI), Sofia Karagiorgou (UBI)	First draft of Executive summary, Introduction, 3.1, 3.11, and 3.12 Sections.
0.3	08/11/2024	Christos Kotronis (UBI), Nikos Androulidakis (8BELLS)	Incorporation of provided contribution for Requirements Traceability Matrix Section from technical partner.
0.4	21/11/2024	Christos Kotronis (UBI), Nikos Androulidakis (8BELLS), Vasileios Lolis (CERTH)	Incorporation of provided contributions (RAIDO components: 3.9 Section) from technical partners.
0.5	29/11/2024	Christos Kotronis (UBI), Yannis Spyridis (KU), Georgios Fragkoulidis (AXON), Andreas Foteas (AXON)	Incorporation of provided contributions (RAIDO components: 3.3, 3.7, 3.8 Sections) from technical partners.
0.6	06/12/2024	Christos Kotronis (UBI), Ioanna Ntinou (QMUL), Yannis Spyridis (KU), Vasileios Lolis (CERTH)	Incorporation of provided contributions (RAIDO components: 3.4, 3.5 Sections) from technical partners.
0.7	12/12/2024	Christos Kotronis (UBI), Ilias Siniosoglou (MINDS), Georgios Tsoumplekas (MINDS), Georgios Tziolas (SID), Georgios Fragkoulidis (AXON)	Incorporation of provided contributions (RAIDO components: 3.2, 3.6, 3.10, 3.13 Sections) from technical partners.
0.8	16/12/2024	Christos Kotronis (UBI), Georgios Theodorou (UBI)	Preparation of final manuscript for internal review and quality control.
0.9	18/12/2024	Ioanna Ntinou (QMUL), Georgios Tzimiropoulos (QMUL)	Internal review and recommendations for improvement.
1.0	23/12/2024	Christos Kotronis (UBI)	Preparation of final version and submission of the deliverable.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
AE	AI Engineer
API	Application Programming Interface
AVE	Adaptive Visualization Engine
CAM	Class Activation Mapping
D	Deliverable
DC	Data Consumer
DE	Data Engineer
DP	Data Provider
DT	Digital Twins
EDA	Exploratory Data Analysis
E2C	Edge-to-Cloud
FR	Functional Requirements
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GRAD-CAM	Gradient-weighted Class Activation Mapping
GUI	Graphical User Interface
HITL	Human-in-the-Loop
IPFS	InterPlanetary File System
LL	Lifelong Learning
LRP	Layer-wise Relevance Propagation
M	Month
ML	Machine Learning
NFR	Non-Functional Requirement
PA	Platform/System Administrator
RAIDO	Reliable AI and Data Optimization
RTM	Requirements Traceability Matrix
SDN	Software-Defined Networking
SHAP	SHapley Additive exPlanations
SLO	Service Level Objective
SOTA	State-of-the-Art
SP	System Property
T	Task
TrL	Trust Level
VAE	Variational Autoencoder
V-L	Vision-Language
WP	Work Package
XAI	Explainable Artificial Intelligence

Disclaimer

Executive summary

The "Design of the General RAIDO Platform and Architecture" delivers and outlines an innovative framework developed under Task 2.3 of the RAIDO workplan to optimize Artificial Intelligence (AI) workflows and datasets across edge and cloud infrastructures. By coordinating diverse computational and communication resources, RAIDO supports energy-efficient AI training and deployment through optimized data management and the incorporation of human-in-the-loop feedback and monitoring solutions to ensure reliability and adaptability. Using an AI-orchestrator paradigm, the architecture enables efficient and seamless deployment of AI models and computing services while enhancing performance, scalability, resource efficiency and optimization.

An essential aspect of the design is the link between stakeholder/user requirements and system specifications—identified as non-functional requirements in previous RAIDO tasks/deliverables—and the functional requirements and technical components of RAIDO, defined in this deliverable. This linkage is demonstrated through a comprehensive Requirements Traceability Matrix, ensuring that every component aligns with real-world demands.

Developed in close collaboration with work packages three (3), four (4), and five (5), the architecture design defines the key components and their interconnections, forming a sustainable, robust, and interconnected RAIDO platform. The platform will support green-AI technologies, advanced data optimization strategies, and serve as the foundation for a fully integrated and impactful RAIDO ecosystem, capable of addressing the evolving demands of Industry 4.0.

1 Introduction

The "Design of the General RAIDO Platform and Architecture" Deliverable (D) 2.3 document represents the essence of Task (T) 2.3, "Overall RAIDO Platform Architectural Design," a critical activity in Work Package (WP) two (2) within the RAIDO project. This task aims to define and develop an innovative, adaptable architecture that supports various Industry 4.0 applications and use cases (smart farming, smart grids, healthcare, and robotics), addressing the diverse challenges of the modern industrial environment. Central to this design is a robust, scalable architecture that integrates a variety of computational and communication resources, Artificial Intelligence (AI) models, and datasets, while focusing on energy-efficient solutions and sustainability.

This document provides a detailed analysis of the RAIDO platform, focusing on the design and integration of its key components, including their logic, interfaces, interactions, and internal mechanisms. It emphasizes the development of green-AI networks, addressing the need for reducing energy consumption while contributing to broader environmental sustainability goals. The architecture is designed to support AI-driven optimization at both the edge and cloud levels, ensuring reliable AI model deployment and data management.

A central feature of this platform is the AI orchestrator, which facilitates the intelligent allocation of resources and network management to optimize performance and system effectiveness. The architecture also integrates human-in-the-loop (HITL) feedback mechanisms, enabling continuous monitoring and improvement of the system's operation.

To ensure the architecture aligns with the needs and goals of all stakeholders, the document also reflects the integration of relevant requirements from previous tasks. A Requirements Traceability Matrix (RTM) is included to map the design to these foundational prerequisites.

This deliverable marks a significant step forward in the RAIDO project, setting the stage for the next phase of development in an increasingly complex and connected Industry 4.0 environment.

1.1 Objectives of the deliverable

The objective of the "Design of the General RAIDO Platform and Architecture" deliverable is to provide a comprehensive and detailed blueprint of the RAIDO platform's architecture, focusing on the design of its core components, their interfaces, internal logic, and interactions. This architecture is developed to ensure high energy efficiency, dynamic scalability, and optimal AI model deployment and data processing. A key aim is to prioritize performance alongside environmental considerations, contributing to sustainability efforts in the context of the global climate crisis.

The deliverable also outlines the approach for orchestrating AI resources and managing network infrastructures to enable intelligent resource allocation and ensure the platform's ability to scale dynamically in response to varying demands. In addition, the architecture supports the development of green-AI technologies, promoting reduced energy consumption during AI model training and deployment, thus aligning with sustainability goals.

Furthermore, the deliverable explores the integration of AI orchestration mechanisms, facilitating the coordination of computational and communication resources across various infrastructure layers. This enables seamless migration of Information Technology (IT) services, supporting the RAIDO platform's adaptability and flexibility in meeting the diverse needs of Industry 4.0 use cases. The overall objective is to lay the foundation for a resilient and efficient platform capable of evolving within the rapidly changing technological landscape.

1.2 Relation with other tasks and deliverables

The current deliverable, D2.3: "Design of the General RAIDO Platform and Architecture" (Month (M) 12) is a direct product of T2.3: "Overall RAIDO Platform Architectural Design", which spans from M03 to M30 within the RAIDO workplan. This task focuses on defining the core architectural components and their interfaces to establish a robust, energy-efficient, and dynamically scalable RAIDO platform. This architecture will coordinate infrastructure to seamlessly support AI models and datasets across diverse computational and communication systems. Central to this task is the collection of functional requirements (FRs), which will inform a comprehensive RTM. This RTM will serve as a foundation for defining business logic, functional specifications, information flow, and the internal architecture of RAIDO's technical components while ensuring a clear understanding of inter-component interactions. T2.3 thus lays the groundwork for a high-level architectural visualization of the RAIDO platform. Moreover, the current deliverable links the user requirements identified and presented in D2.1: "Report on the Specifications of Pilots, Specifications, KPIs, Requirements & Novel Enabling Technologies", as well as the technical roles of actors and the system specifications—here, identified as non-functional requirements (NFRs)—with the FRs and the RAIDO components, ensuring alignment between user needs and technical implementation.

T2.3 receives D1.1: "Project Management Handbook: Administration, Technical, Data, Business, and Dissemination" as input and produces D2.3 (current) and D2.4: "Final RAIDO Platform and Architecture" (M30) as outputs.

1.3 Document structure

This document is designed to guide the reader from a high-level understanding of the RAIDO architecture to an in-depth exploration of each architectural component and its function and interaction within the larger system. It provides a comprehensive overview

of the RAIDO reference architecture, componentization, and implementation strategies. The structure is as follows:

- Section 1 (“*Introduction”*) introduces the objectives of the deliverable, outlines its relation to other tasks and deliverables, and presents the document’s structure.
- Section 2 (“*Architecture componentization”*) provides an overview of the RAIDO architecture, detailing the reference architecture and defined components, along with a walkthrough of this and the involved actors.
- Section 3 (“*RAIDO components”*) delves into the specific technical components of RAIDO. It covers: core AI-Pipelines and their optimization, data enrichment, curation, and federated mining techniques, synthetic data generation methods, advanced adaptation and distillation methods for vision-language models, life-long learning within federated architectures, data lake infrastructure and management, frameworks focused on AI explainability, benchmarking and feedback mechanisms for progress monitoring, adaptive AI processes and visualization tools, green-AI orchestration and self-evolving mechanisms, network management capabilities and resource allocation and deployment across edge-to-cloud (E2C) continuums, and blockchain and smart contract integration with InterPlanetary File System (IPFS) support. Each component includes subsections on *business logic*, *functional description*, *information flow and internal architecture*, and *component interactions*.
- Section 4 contains the “*Requirements Traceability Matrix”*, mapping user requirements and NFRs, identified in a previous deliverable (D2.1) to respective FRs and components for ease of reference and tracking.
- Section 5 concludes the document with a summary of findings and directions for future work.
- Section 6 lists the *“References”* consulted and cited throughout the document, supporting the research and development insights presented.

2 Architecture componentization

2.1 RAIDO high-level architecture and defined components

The RAIDO architecture serves as a comprehensive, end-to-end, AI-centric framework designed to facilitate the generation, training, evaluation, and optimization of AI models through a well-orchestrated pipeline of technical components. Its modular design ensures a comprehensive workflow, from loading user-provided datasets and AI models to delivering performance insights, resource metrics, and actionable feedback. RAIDO's technical components are organized into interconnected tasks (outlined in the RAIDO workplan), each having a specific technical role in achieving performance, energy efficiency, and adaptability.

At a high level, the user interacts with RAIDO by providing datasets, synthetic data requests, or pre-trained AI models while defining service-level objectives (SLOs) such as accuracy, energy consumption, or bias constraints. RAIDO automatically processes these inputs across its pipelines, combining advanced AI and Machine Learning (ML) techniques, federated learning, orchestration tools, eXplainable AI (XAI), and continuous feedback mechanisms to deliver optimized results.

As illustrated in the following Figure 1, we pass through the architecture from top to bottom and left to right, starting with the user at the top. The user's data and models flow systematically and logically through RAIDO's core components, including data enrichment, synthetic data generation, few-shot learning, federated learning, or green-AI model training, and resource monitoring and orchestration, resulting in optimized models, performance and energy metrics, and actionable recommendations for the user.

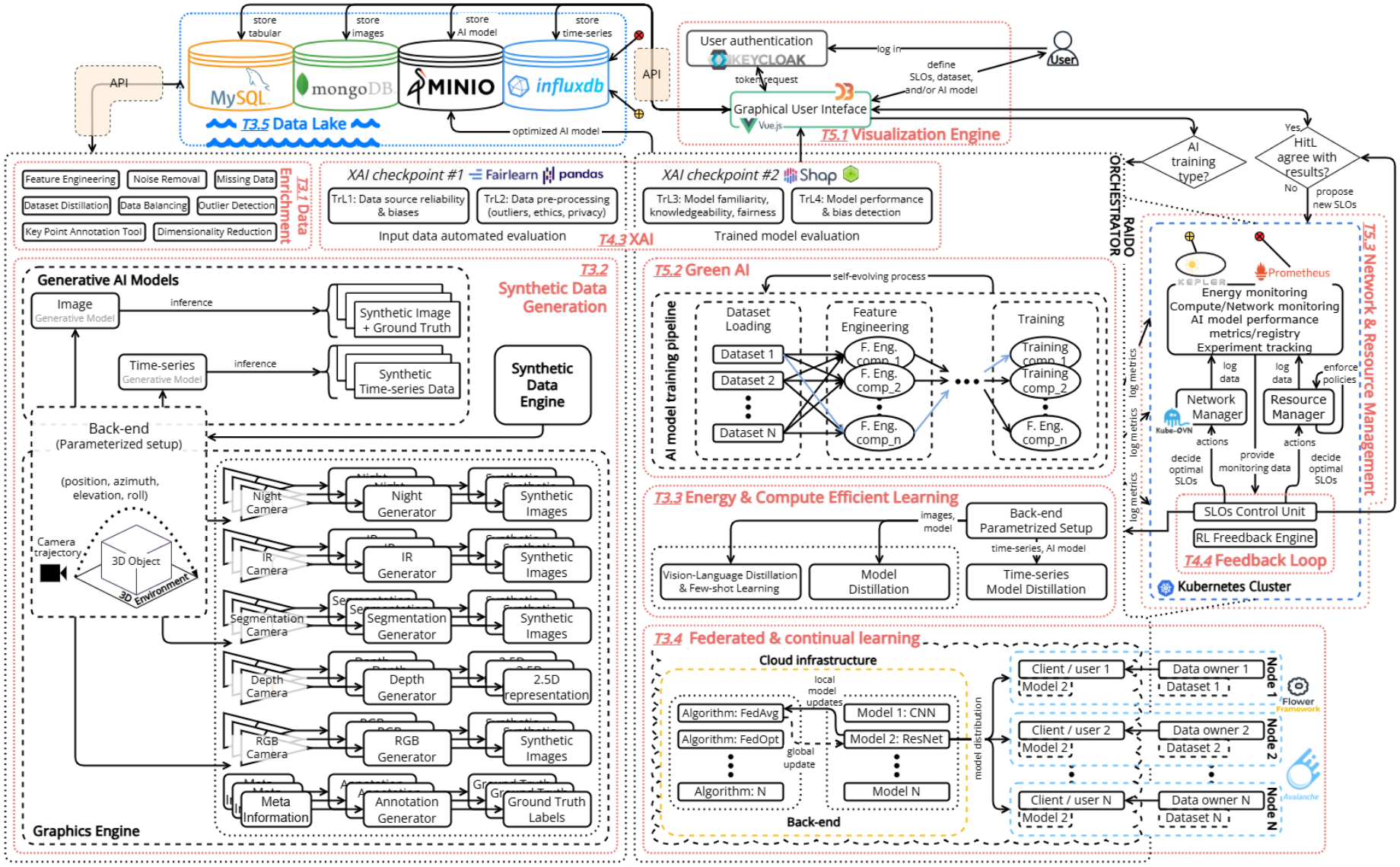

Figure 1. RAIDO Reference Architecture.

The RAIDO workflow begins with the user, who interacts with the *Visualization Engine*. This provides an intuitive Graphical User Interface (GUI) to input datasets and/or AI models, define SLOs, monitor workflows, and evaluate results such as performance metrics, energy usage, and other model and dataset insights. For example, a user can monitor model accuracy over multiple training epochs while visualizing energy consumption trends. Secure login is ensured through Keycloak-based user authentication.

The *Data Lake* serves as the centralized storage for all user-input data, synthetic outputs, trained models, and monitoring logs. It integrates tools like MySQL, MongoDB, MinIO, and InfluxDB to handle diverse data formats (e.g., tabular data, time-series data, trained AI models, and images).

User-provided datasets are ingested and prepared for further processing via *Data Enrichment*. This task focuses on validating data quality, identifying biases, and applying necessary preprocessing steps such as data cleansing and normalization. RAIDO ensures that input data meets reliability standards via *XAI* using tools such as Fairlearn and Pandas (XAI Checkpoint #1). For example, a user may upload a dataset containing imbalanced samples (e.g., biased labels for image classification). The enrichment step identifies these issues and recommends data rebalancing strategies, ensuring the dataset is fit for model training. In this component, Trust Levels (TrLs) are introduced as a measure of data reliability and fairness.

Next comes the *Synthetic Data Generation* component, which addresses scenarios where real-world data is insufficient, biased, or unavailable. RAIDO leverages Generative AI Models and a Graphics Engine to create high-fidelity synthetic datasets. Its Back-End Parameterized Setup enables the simulation of realistic environments, combining 3D object models, camera trajectories, and environmental parameters such as azimuth, elevation, and lighting conditions. Multiple camera types—Red, Green, and Blue (RGB), Night, Infrared (IR), Segmentation, and Depth—capture synthetic data, generating outputs like Synthetic Images, Ground Truth Labels, and/or 2.5D Representations.

To address data privacy and decentralized data challenges, RAIDO implements *Federated Learning* (FL). This component enables collaborative model training across distributed datasets owned by multiple clients/users. Client Nodes train local model updates using their private datasets. RAIDO aggregates these local model updates via using specialized algorithms (e.g., FedAvg [1] or FedOpt [2]) into a globally updated optimized model. For example, in the healthcare domain, hospitals training an AI model for disease detection can collaboratively contribute local updates without sharing sensitive patient data. In addition, the Continual Learning capability allows AI models to incrementally learn from new data as it becomes available, ensuring sustained performance while preserving previously learned knowledge and minimizing the risk of catastrophic forgetting.

The user can choose AI model training type via the GUI. When real-world training data is limited, RAIDO employs *Few- and Zero-Shot Learning* to enable AI models to generalize to new tasks with minimal or no labelled examples. By leveraging *vision-language distillation*, RAIDO transfers knowledge from large pre-trained models to new tasks, reducing the need for extensive labelled data. In addition, Model Distillation reduces the size and complexity of large AI models by compressing their parameters, converting them into lighter alternatives suitable for deployment on resource-constrained devices. This task ensures that users can achieve high-performing AI models even in data-scarce environments.

RAIDO also incorporates the *Green-AI* component to optimize both the energy efficiency and performance of AI models during training and deployment. It focuses on minimizing resource consumption through techniques like energy-efficient hyperparameter tuning, model distillation, and the continuous optimization of training pipelines. By integrating Green-AI principles, RAIDO ensures that AI models can be trained effectively without excessive energy or resource usage, thereby reducing the environmental impact of computationally intensive tasks. In addition, Green-AI integrates with the Self-Evolving Data Orchestration flow, which adapts AI workflows in real-time, ensuring that resources are allocated dynamically to meet energy and performance goals.

The RAIDO Orchestrator, powered by Kubernetes, acts as the brain behind the operation, bringing together all the AI model training components and ensuring they work in harmony to train the model efficiently. This coordinates workflows across its components. It ensures efficient scaling, monitoring, scheduling, and execution of processes while optimizing compute and network resources through tools like Prometheus or Kepler. RAIDO integrates a Network Manager and Resource Manager to track system performance metrics (energy consumption, Central Processing Unit (CPU)/Random Access Memory (RAM), bandwidth, etc.) and enforce optimization policies. If a training job exceeds user-defined SLOs (e.g., high resource consumption), RAIDO can dynamically adjust configurations, like scaling down compute and/or network resources or proposing alternative training parameters. To provide an example, during a training job, the orchestrator loads datasets from the Data Lake, triggers the Synthetic Data Engine, and executes the training pipeline while logging performance metrics. This centralized storage enables RAIDO to efficiently manage large-scale data and model updates while supporting real-time analytics and resource optimization.

After a model has been trained and optimized, XAI Checkpoint #2, utilizing tools such as SHapley Additive exPlanations (SHAP) or Lime, evaluates the AI model during the training and validation stages. This checkpoint ensures and verifies that the model is both trustworthy and explainable (via TrLs), addressing fairness, bias, and performance issues, while it identifies and mitigates any biases or errors that might affect decision-making.

RAIDO also employs a Reinforcement Learning (RL) *Feedback Loop* to continuously improve processes. It evaluates the performance of models and resource usage. When objectives are not met, RAIDO autonomously proposes refinements, such as tuning hyperparameters or improving synthetic data parameters. For instance, if a model's bias detection fails at an XAI checkpoint, RAIDO provides recommendations to refine data preprocessing or model architecture, ensuring compliance with fairness and performance standards.

To further enhance transparency, security, and traceability, RAIDO integrates *Blockchain* technology into its architecture. Blockchain provides an immutable ledger for tracking data transactions and model updates across distributed environments. This integration ensures that model training can be audited for consistency and integrity.

To simply illustrate RAIDO's workflow, consider the following end-to-end scenario:

1. Input: A user uploads a real-world dataset containing limited edge-case images (e.g., nighttime scenes) and defines SLOs like model accuracy and energy efficiency.
2. Synthetic Data Generation: RAIDO generates additional synthetic data using its Graphics Engine, supplementing the dataset with diverse nighttime images.
3. Training and Evaluation: The user trains an AI model using RAIDO's Few-/0-Shot Learning pipeline and Federated Learning if privacy concerns exist. The model is evaluated for accuracy, bias, and energy consumption using XAI checkpoints.
4. Resource Monitoring and Feedback: RAIDO monitors energy consumption and resource usage. If inefficiencies are detected, it proposes refinements via the RL Feedback Loop.
5. Output: The user receives a trained, optimized model alongside detailed performance insights, energy metrics, and recommendations for further improvement.

Based on all these components, RAIDO can provide an adaptable, efficient, and user-driven AI development environment that balances performance, resource optimization, and fairness.

2.2 Architecture walkthrough & RAIDO actors

In previous deliverable D2.1, a wide range of stakeholders for each of the four (4) RAIDO pilots was identified and outlined, classifying them based on their interactions, functions, and responsibilities within the RAIDO framework. These roles were categorized in terms of system usage, data/insight consumption, data provision, or the oversight and/or management of AI-driven data processes. Specifically, technical RAIDO actors or user roles such as End-User, Data Consumer (for brevity, it will be abbreviated as DC), Data Provider (for brevity, DP), Data Engineer (for brevity, DE), AI Engineer (for brevity, AE), and Platform Administrator (for brevity, PA), were defined.

In this deliverable, we map these technical roles to the technical components of the RAIDO platform, demonstrating how each role interacts with the platform's core features and functionalities. AEs focus on AI model optimization, green-AI initiatives, and model learning. DEs handle data management, enrichment, and preparation for AI models, while End-Users/DCs interact with the system to retrieve results, gain insights, and make decisions. PAs ensure the infrastructure and resources are efficiently optimized and managed across the platform, enabling smooth interaction between components. Table 1 below shows the actors-components linkage in detail.

Table 1. Mapping technical RAIDO actors with RAIDO components.

Technical role	RAIDO component	Description
AI Engineer (AE)	Core Optimized AI-Pipelines	Responsible for designing and optimizing AI pipelines for specific tasks (e.g., energy optimization, management, etc.), ensuring these pipelines are effective and efficient.
	Self-evolving Green-AI and Data Orchestration	Develops AI models that adapt to energy optimization and resource management, contributing to the green-AI component by enhancing performance and reducing energy consumption during AI model training and deployment.
	Life-Long Learning in Federated Architectures	Designs AI models that can continuously learn and evolve in decentralized, federated environments, improving RAIDO's adaptability and performance over time.
	Few- & Zero-Shot Adaptation and Distillation of Vision-Language Models	Develops techniques that allow RAIDO to operate with minimal data for tasks like visual recognition and natural language processing, enhancing its flexibility and scalability.
	Frameworks for AI Explainability	Contributes to the creation of transparent AI models that help stakeholders/end-users understand and make decisions, fostering trust and interpretability, especially for energy optimization algorithms.
Data Engineer (DE)	Data Enrichment, Curation, Distillation, and Federated Mining	Prepares and preprocesses data for AI models, ensuring it is accurate, comprehensive, and ready for analysis. The engineer also facilitates federated data mining to improve model performance across distributed environments.
	Synthetic Data Generation	Generates synthetic datasets to augment real-world data, allowing AI models to be trained on diverse and varied datasets, especially when real-world data is limited or difficult to obtain.
	Data Lake	Manages the data lake, ensuring efficient storage, access, and management of large volumes of structured, semi-structured, and unstructured data. This data is critical for training AI models and optimizing system performance.
	Adaptive AI Processes and Visualization Engine	Interacts with the visualization engine to monitor metrics (e.g., energy consumption or

Technical role	RAIDO component	Description
End-User/ Data Consumer (DC)		performance) and insights for AI models, datasets, etc., enabling informed decision-making.
	Reinforced Benchmarking and Feedback-Based Progress Monitoring	Provides feedback based on RAIDO outputs, helping refine AI models and ensuring continuous improvement in the system's performance and accuracy.
Platform Administrator (PA)	Networking Management	Manages network metrics to inform AI models for Service Level Objective (SLO) adaptations.
	AI-based E2C Continuum Resource Allocation & Deployment	Ensures that resources are effectively distributed across edge, cloud, and continuum systems based on AI-driven insights, maintaining optimal system performance and scalability.
	Blockchain & Smart Contracts with IPFS Support	Manages deployment of blockchain and smart contract systems, ensuring that the RAIDO platform remains secure and efficient for all transactions and data exchanges.

3 RAIDO components

In this Section, we delve into each technical component of the RAIDO architecture, presenting them in alignment with the RAIDO workplan, i.e. we begin with “T2.2: Core Optimized AI-Pipelines” and proceed through each task in sequence (i.e. “T3.1 Data enrichment, curation and distillation, and Federated Mining”, “T3.2 Synthetic Data Generation”, “T3.3 Few- & Zero-Shot adaptation and distillation of Vision-Language Models”, “T3.4 Life-Long Learning in Federated Architectures”, “T3.5 Data Lake”, “T4.3 Frameworks for AI explainability”, “T4.4. Reinforced benchmarking and feedback-based progress monitoring”, “T5.1 Adaptive AI processes and visualization engine”, “T5.2 Self-evolving green-AI and data orchestration”, “T5.3 Networking management and AI-based E2C continuum resource allocation & deployment”), ultimately reaching “T5.4: Blockchain & Smart Contracts with IPFS Support”, covering the business logic of all the components, their functional description, internal architecture and information flows, and interactions with other components.

3.1 Core Optimized AI-Pipelines

3.1.1 Business logic

The business logic behind the AI pipeline is designed to optimize the balance between edge computing efficiency and centralized model intelligence for scalability, performance, fairness, and transparency. By leveraging edge nodes for localized data processing, validation, and model training, the system reduces latency and optimizes resource utilization, while maintaining flexibility across diverse environments. Centralized processing aggregates and evaluates models from the edge, ensuring that performance is continuously monitored, and biases are identified and corrected to promote fairness. Ethical AI principles, such as privacy-preserving mechanisms and secure data storage, are integral to every phase, protecting user data and ensuring compliance. The incorporation of explainability tools like SHAP, LIME, or Grad-CAM helps gain trust from stakeholders by providing transparent, interpretable predictions. Through continuous monitoring and model updates, the system adapts to changing data, ensuring that deployed models remain effective and reliable in real-world scenarios. This holistic approach aligns with business objectives of creating efficient, ethical, and trustworthy machine learning systems that can scale seamlessly across edge and centralized platforms.

Outputs from this component will influence all other technical work packages and tasks of RAIDO (WP3: “Data Enrichment Solutions & Optimized Trustworthy AI Architectures,” WP4: “Human-centred AI,” and WP5: “Energy efficient E2C deployment and Green AI Orchestration”), ensuring a unified pipeline for data generation and AI optimization.

3.1.2 Functional description

The following Table 2 provides a clear, structured summary of the functionality and characteristics of the AI-Pipelines component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs & pre-conditions*, *outputs & post-conditions*, any Application Programming Interface (*API*) used to interact with other components, and the *technology* used to implement it.

Table 2. AI-Pipelines component: Functional description.

Component ID	C01
Component name	Core Optimized AI-Pipelines
Sub-name	AI-Pipelines
Description	The ML pipeline consists of modular, well-defined stages that handle data processing, model training, evaluation, and deployment across edge and centralized nodes. It includes support for data distillation, lifelong learning, transfer learning, and few-shot learning. The pipeline integrates ethical AI principles and XAI mechanisms to ensure transparency, fairness, and energy-efficient AI development.
Inputs & pre-conditions	• Raw data collected at edge nodes. • Labeled/unlabeled datasets for training. • Configuration for AI model optimization parameters.
Outputs & post-conditions	• Optimized and trained ML models. • Aggregated and evaluated global models. • Deployed models for inference at edge or centralized nodes. • Insights into model bias, performance metrics, and explainability reports. • Secure storage of data and models.
API	Under development
Implementation technology	Programming language: Python

3.1.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 2.

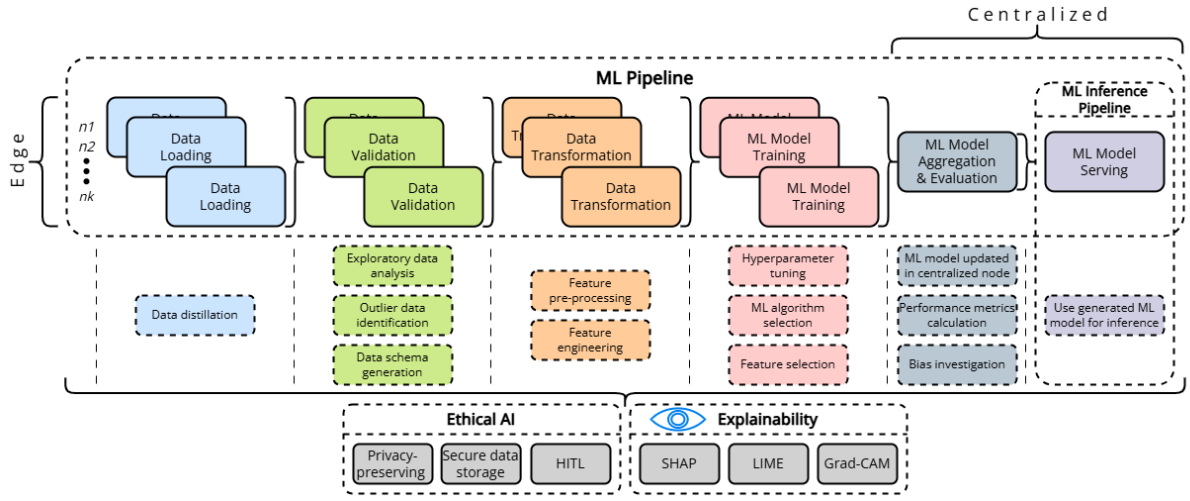

Figure 2. AI-Pipelines component: Internal architecture.

Regarding the information flow:

- Raw data is collected and processed in edge nodes (n1, n2, ..., nk).
- The pipeline starts with Data Loading, followed by Validation and Transformation.
- ML models are trained locally on processed data using selected algorithms.
- Trained models from edge nodes are aggregated and evaluated centrally.
- Performance metrics are calculated, and model biases are investigated.
- Final models are updated centrally and served back to the edge or for global inference.
- Once the ML model is deployed, it is used to make predictions on new data, ensuring scalability across edge and centralized systems.

Regarding the internal architecture of the AI-Pipelines component, as depicted in Figure 2, there are two (2) key layers: (i) the Edge layer that handles data preprocessing, validation, and local model training, and allows distributed processing to optimize resource usage and reduce latency, and (ii) the Centralized layer, which aggregates edge models into a robust unified model, provides tools for evaluation, bias correction, and performance monitoring, and deploys ML models to inference pipelines for use in production. There are also Ethical AI principles to ensure privacy and security in every phase of the pipeline, while explainability tools enhance trust by offering interpretable predictions. Delving into more details, the ML pipeline consists of modular, well-defined stages:

- Edge processing
    1. Data loading (blue-colored boxes):
        - Distributed data collection across multiple edge nodes *n1* to *nk*.
        - Preparation of raw data for downstream ML pipeline processes.
        - Data distillation for compression or data refinement.
    2. Data validation (green-colored boxes):
        - Exploratory data analysis to understand and summarize data properties.
        - Outlier data identification to detect and address anomalies in edge datasets.
        - Data schema generation to standardize data formats for consistency across nodes.
    3. Data transformation (orange-colored boxes):
        - Feature pre-processing to clean, normalize, and prepare data.
        - Feature engineering to generate new features to improve ML performance.
    4. ML model training (pink-colored boxes):
        - Hyperparameter tuning to optimize model configurations for better performance.

        - Algorithm selection to choose the best ML algorithms suited for tasks.
        - Feature selection to reduce feature dimensionality for efficient learning.
- Centralized Processing
    1. Model aggregation & evaluation (grey-colored boxes):
        - Aggregates models trained on edge nodes into a unified model.
        - Performance metrics calculation to measure model accuracy, efficiency, and robustness.
        - Bias investigation to ensure fairness and address any model biases.
        - ML models are updated centrally for distribution back to edge nodes.
    2. ML inference pipeline (purple-colored boxes):
        - Model serving to deploy the final model for real-world inference tasks.
        - Generated ML models can be used at both edge and centralized locations.
- Cross-Functional Components
    1. Ethical AI:
        - Privacy-preserving mechanisms.
        - Secure data storage solutions.
        - HITL systems for oversight.
    2. Explainability:
        - Tools such as SHAP, LIME, and Grad-CAM to explain and interpret model predictions.

This combined system enables a scalable, ethical, and explainable ML pipeline, balancing edge computing efficiency with centralized intelligence for optimal model performance and trustworthiness.

3.1.4 Interaction with other components

The AI-Pipelines component interacts with the components listed in Table 3 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 3. AI-Pipelines component: Interactions.

Interacting Component	Description
Network & Resource Managers	It can interact with Network & Resource Management to ensure that resources (compute power, bandwidth) are optimized across both Edge Processing and Centralized Processing stages. This interaction is crucial for energy-efficient AI and minimizing computational costs, especially for large-scale AI training and model deployment.

Green-AI	It can interact with Green-AI to apply energy-efficient techniques like data distillation, model distillation, and hyperparameter tuning to reduce the environmental impact of large-scale AI training processes.
Data Lake	The component interacts with the Data Lake for data storage, retrieval, and management. The Data Lake serves as the repository for raw data, processed data, and trained models, ensuring that the data pipeline is efficient and well-organized.

3.2 Data Enrichment, Curation, Distillation, and Federated Mining

3.2.1 Business logic

The Data Enrichment, Curation, Distillation, and Federated Mining component of the RAIDO architecture is responsible for optimizing data used within the RAIDO platform. This data optimization process encompasses functionalities related to data curation, enrichment, distillation, and federated data mining. By employing these techniques, the component ensures that the data is appropriately formatted for use in the model optimization pipeline of the AI models. Furthermore, the component produces high-quality data to improve downstream performance and eliminates redundancies, thereby maintaining high-performance levels while reducing computational and energy demands required by the AI models that utilize this data.

One of the primary modules of this component is the Data Enrichment, Curation, and Distillation module. This module focuses on enhancing both the quality and the compactness of the provided raw data. It addresses common imperfections found in most real-world datasets by handling missing data, detecting outliers, and removing noise signals. In addition, the module features various data enrichment and transformation methods to facilitate the effective use of this data in training and evaluating AI models. To further improve efficiency, it employs techniques like dataset distillation to minimize redundancies in the data by creating a compact set of fewer representative data points.

The second main module of the component is the Federated Data Mining module, which addresses dataset imbalances while preserving their privacy. Specifically, it leverages other RAIDO users' data to identify available suitable samples for addressing imbalances in the examined dataset. However, if there are restrictions that prevent access to other users' data, or if no such data is available, the module also gathers the necessary relevant information and metadata. This information is subsequently passed to the Synthetic Data Generation component through the Orchestrator, enabling the generation of synthetic samples tailored to tackle the specific imbalance.

3.2.2 Functional description

The following Table 4 provides a clear, structured summary of the functionality and characteristics of the Data Enrichment component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 4. Data Enrichment component: Functional description.

Component ID	C02
Component name	Data Enrichment, Curation, Distillation, and Federated Mining
Sub-name	Data Enrichment
Description	The core functionality of this component is to generate high-quality, pre-processed data suitable for optimizing AI model performance. Specifically, it takes raw data as input and processes it through two (2) main subcomponents: Data curation, enrichment, and distillation create high-quality, curated data based on the Orchestrator's requests. Federated data mining addresses dataset imbalances by leveraging useful data from other users while preserving privacy. If other users' data is unavailable or restricted, this subcomponent provides necessary information for the Synthetic Data Generator component to create new data samples. Finally, the pre-processed data is stored in the Data Lake for use in the subsequent model optimization pipeline, while relevant metadata is sent back to the Orchestrator.
Inputs & pre-conditions	i. Images, time series, tabular data, text ii. Availability of data iii. User's consent
Outputs & post-conditions	i. Transformed data (i.e. pre-processed, enriched, curated, distilled) ii. JSON file containing results
API	Under development
Implementation technology	Programming language: Python Frameworks/Libraries: Docker, skimage, Pillow, sklearn, pandas, FastAPI, flask

3.2.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 3.

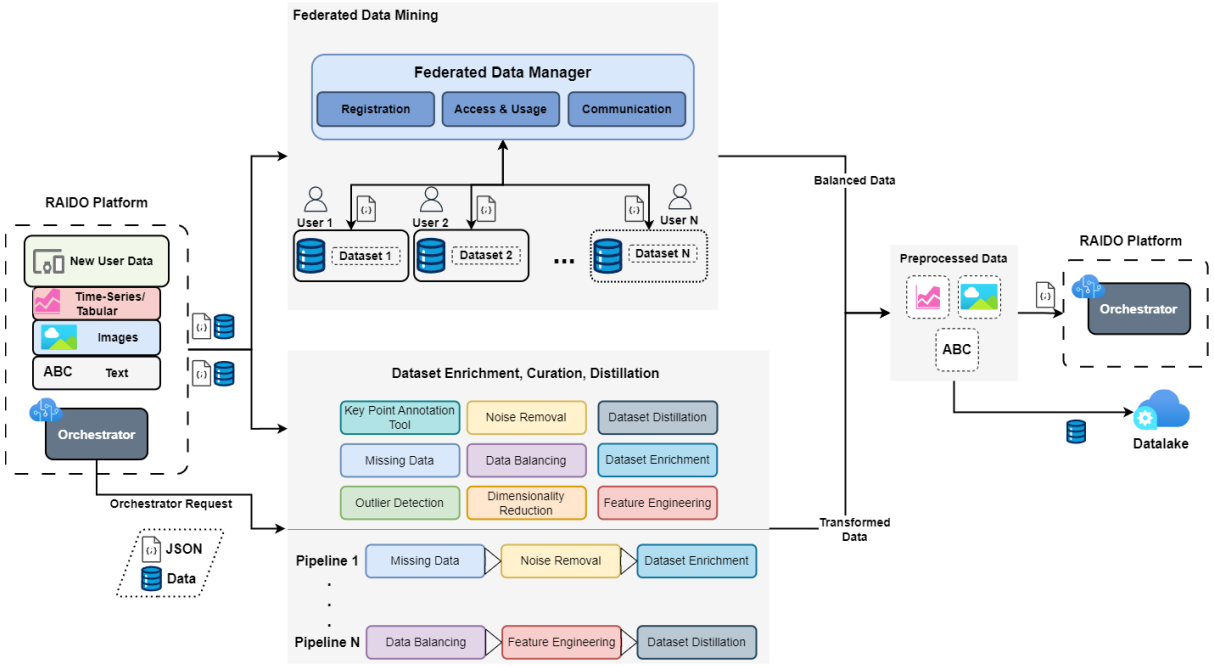

Figure 3. Data Enrichment component: Internal architecture.

The architecture diagram illustrates the information flow of this component and its interactions with other RAIDO components.

As part of RAIDO's core functionality, the component communicates directly with the Orchestrator, which is responsible for sending specific requests to it. These requests activate the appropriate modules within the component and provide them with the necessary information and parameters to perform their designated tasks. The component comprises two (2) hierarchically equivalent (i.e., there is no predefined order of execution) subcomponents, i.e. the Dataset Enrichment, Curation, and Distillation, and the Federated Data Mining. Consequently, the Orchestrator decides which module to trigger and in what sequence based on the specific requirements of the pipeline. The same principle also applies at a more fine-grained level within the Dataset Enrichment, Curation, and Distillation module, which consists of nine (9) grouped functionalities, each offering multiple methods. In this context, the Orchestrator determines which specific functionalities will be applied and in what order as part of the overall data and AI model optimization pipeline of RAIDO.

In addition, since the functionalities of this component operate directly on datasets, the availability of data on which the pre-processing will be performed is crucial. Therefore, the component is also directly linked to RAIDO's Data Lake, from which it retrieves the datasets on which it operates.

Within the component, the Dataset Enrichment, Curation, and Distillation module focuses on improving data quality and compactness by handling issues such as noise, missing data, and redundancies, while the Federated Data Mining module addresses any imbalances in the original data by leveraging additional data or metadata. The outcome of these processes includes transformed, balanced, and pre-processed data, which is stored in the Data Lake for future use by the same or different optimization pipelines, and

associated metadata, documenting the completed procedures, which is returned to the Orchestrator, providing the detailed results of the completed tasks and can be used by the Orchestrator in subsequent pipeline steps.

3.2.4 Interaction with other components

The Data Enrichment component interacts with the components listed in Table 5 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 5. Data Enrichment component: Interactions.

Interacting Component	Description
RAIDO Orchestrator	The component functions as a responsive module within RAIDO, receiving requests from the Orchestrator with the aim of enhancing both the quality and quantity of data. To accomplish this, the component dynamically selects appropriate methods from a pool of functionalities, including data enrichment, feature engineering, noise removal, outlier detection, handling of missing data, dataset distillation, and federated data mining. These methods work together to improve the available data. Subsequently, once the preprocessing tasks are completed, any metadata linked to the executed data pipeline is returned to the Orchestrator, allowing it to assess the current state of the overall data and AI model optimization process and to plan its next steps in the optimization pipeline.
Data Lake	The component also interacts directly with the Data Lake component before and after performing its internal functions. Specifically, before preprocessing, the component retrieves the raw data requested by the Orchestrator from the Data Lake to apply the designated preprocessing steps. After preprocessing, the transformed data is stored back into the Data Lake for use by subsequent components in the optimization pipeline.

3.3 Synthetic Data Generation

3.3.1 Business logic

The Synthetic Data Generation component plays a central role in managing synthetic data generation processes to overcome limitations in real-world datasets. At its core, the engine coordinates requests for synthetic data generation coming from the RAIDO Orchestrator and directs them to the most suitable subsystem —either the Graphics Engine or the Generative AI Models— based on specific task requirements. This ensures efficient resource utilization and targeted data generation, enabling the RAIDO platform to address specific challenges such as data balancing and augmentation.

The Graphics Engine and Generative AI Models are designed to complement each other, forming a reliable, robust pipeline that enhances RAIDO's data enrichment capabilities

(cf. Section 3.2). The Graphics Engine features a parameterized simulation environment, which allows precise control over variables and conditions such as lighting, angles, and object placement, generating photorealistic outputs with detailed annotations tailored to various use cases. Meanwhile, the Generative AI Models leverage advanced architectures—including Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion models—to produce realistic and diverse synthetic data, including images and time-series data. Together, these subsystems enable the engine to create scalable and richly annotated synthetic datasets that meet the demands of downstream applications while maintaining adaptability and precision in data generation. This synergy ensures that the Synthetic Data Generation component effectively addresses the scarcity of real-world data and enriches the RAIDO platform to improve the performance and robustness of AI models across its various pilots.

3.3.2 Functional description

The following Table 6 provides a clear, structured summary of the functionality and characteristics of the Synthetic Data Generation component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 6. Synthetic Data Generation component: Functional description.

Component ID	C03
Component name	Synthetic Data Generation
Sub-name	Synthetic Data Generation
Description	The core functionality of this component is to generate high-quality synthetic data and associated labeling, supporting multimodal data types such as images and time-series. It leverages generative models and digital twins through graphic engines to achieve this. The component processes pilot data as input, including image and time-series data, and routes it through a parameterized synthetic data engine. Based on specific requirements, it either employs the graphics engine to simulate 3D environments with precise control over object positioning, lighting, and camera trajectories or utilizes generative AI models to produce synthetic images and temporal time-series data. This approach ensures the output mimics real-world patterns with high fidelity while providing robust annotations and depth maps where necessary.
Inputs & pre-conditions	i. Images, time-series data ii. Availability of computational resources
Outputs & post-conditions	Synthetic data (images, time-series data, depth maps) and/or ground-truth labels
API	Under development
Implementation technology	Programming language: Python Frameworks/Libraries: Docker, Flask Graphics engine/Simulation tool: Unity

3.3.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 4.

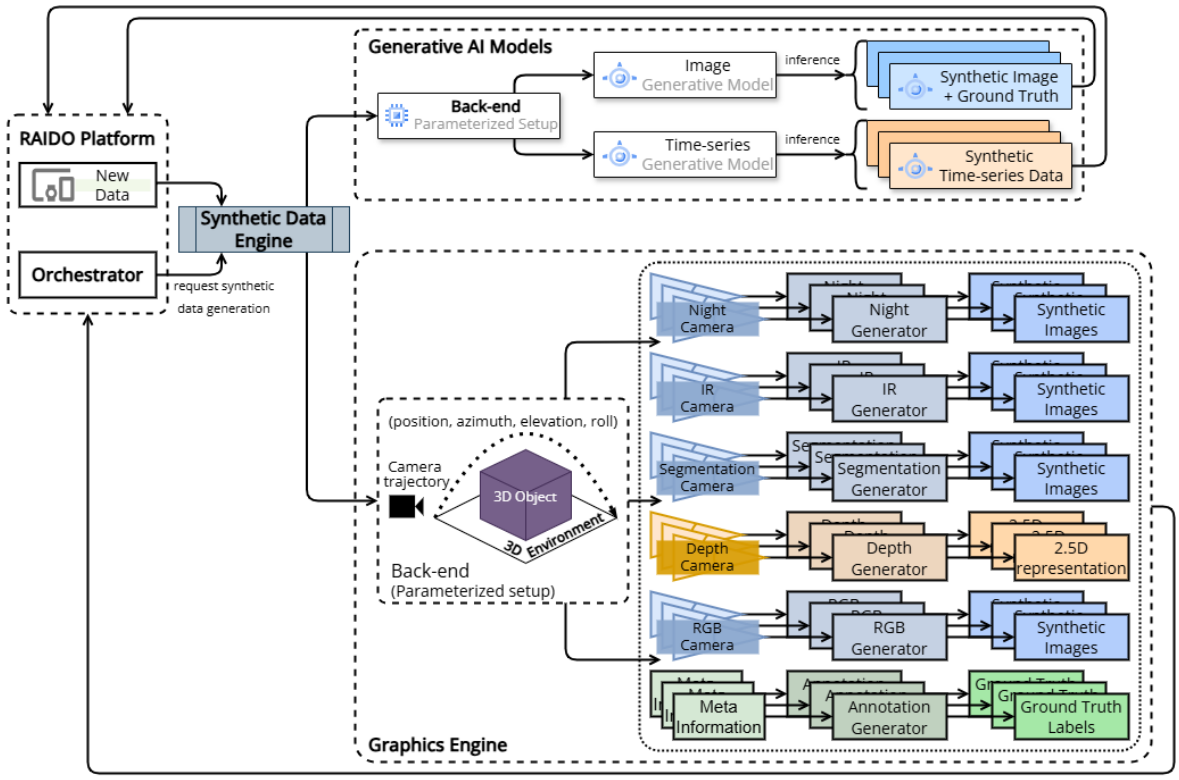

Figure 4. Synthetic Data Generation component: Internal architecture.

This internal architecture illustrates the Synthetic Data Generation component, which combines advanced graphics-based simulation techniques with state-of-the-art (SOTA) generative AI models to produce a wide diverse range of synthetic datasets. At its core, the RAIDO Orchestrator within the RAIDO Platform manages pipelines that request the generation of synthetic data and forwards these requests to the Synthetic Data Engine. This engine acts as an interface, directing the requests either to the Graphics Engine or the Generative AI Models, depending on the type of data required.

The Graphics Engine operates as a parameterized simulation environment capable of replicating complex real-world scenarios. At its core, a backend system renders 3D environments by manipulating parameters such as object positions, camera trajectories, and settings like azimuth, elevation, and roll. Using this environment, the Graphics Engine generates a variety of outputs tailored to various use cases. These outputs include synthetic nighttime imagery, infrared imagery, segmentation masks, depth data for 2.5D representations, standard RGB images, and annotations with ground-truth labels. The Graphics Engine provides advanced photorealism and precise control over environmental conditions. This capability is particularly beneficial in scenarios where real-world data is unavailable, scarce, imbalanced, or difficult to collect, such as rare events or low-resource cases.

In parallel, the Generative AI Models subsystem provides a complementary approach to synthetic data generation. This subsystem enables users to either fine-tune various types of generative models on uploaded input data or leverage pre-trained, recommended models designed to efficiently produce realistic, high-quality synthetic data. The fine-tuning process adapts the generative models to the specific characteristics of the input data, ensuring that the synthetic outputs align closely with the desired parameters and use cases. Once trained or fine-tuned, the models can infer synthetic outputs, including images paired with ground-truth labels, and time-series data for applications requiring dynamic or temporal patterns. The image generative models specialize in creating realistic synthetic imagery, while the time-series generative models focus on producing sequences of data that simulate real-world temporal patterns. Underlying these generative models are advanced deep learning techniques, including architectures like GANs, VAEs, or Diffusion-based models, depending on data types and the complexity of the task. These techniques enable the models to learn and reproduce intricate patterns in the training data, resulting in synthetic outputs that closely resemble real-world distributions. This facilitates the training of AI models with large volumes of high-quality synthetic data. Once the synthetic data is generated, it is returned to the Orchestrator, where it is utilized for downstream applications, enhancing AI model performance, robustness, and scalability.

3.3.4 Interaction with other components

The Synthetic Data Generation component interacts with the components listed in Table 7 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 7. Synthetic Data Generation component: Interactions.

Interacting Component	Description
RAIDO Orchestrator	The Synthetic Data Generation engine operates as a responsive component within the RAIDO Platform, receiving data requests from the Orchestrator to address specific needs, such as dataset balancing or augmentation. Upon receiving these requests, the engine dynamically selects the appropriate subsystem —either the Graphics Engine or the Generative AI Models— to generate targeted synthetic datasets. This streamlined interaction ensures the delivery of high-quality, application-specific data to the Orchestrator, supporting downstream pipelines, and enhancing RAIDO's capability to create robust datasets for diverse use cases.

3.4 Few- & Zero-Shot adaptation and distillation of Vision-Language Models

3.4.1 Business logic

This component, also titled Energy and Compute Efficient Learning, focuses on adapting models to downstream tasks with minimal data and energy consumption. It employs techniques like model/knowledge distillation, Vision-Language (V-L) distillation, and soft prompt learning to enable efficient fine-tuning and model adaptation. Integrated with the RAIDO Orchestrator, the component processes training data and models, routing them to the most suitable subsystem —model/knowledge distillation, prompt learning, or a combination of both— to ensure energy-efficient training. In addition, it incorporates knowledge distillation methods for time-series data, broadening its applicability across diverse tasks.

The component will leverage SOTA technologies to achieve energy- and data-efficient training for both vision models and V-L models, which combine visual and textual data to understand and generate multimodal content, such as CLIP [1] . For vision models, the component will utilize Decoupled Knowledge Distillation (DKD) [2] , a technique that separates logit and feature distillation into independent optimization processes. This method enables effective knowledge transfer from large teacher models to smaller, more efficient student models. For V-L models, the component will use TinyCLIP [3] , which combines affinity mimicking and weight inheritance to distill large-scale language-image models into smaller student models, while maintaining performance. To support soft prompt learning, the component will incorporate methods like Language-Aware Soft Prompting (LASP) [4] that enhance adaptation and mitigate overfitting using textual prompts. For time-series forecasting tasks, SOTA time-series models such as PatchTST [5] , combined with distillation losses will be employed to improve both performance and efficiency.

These technologies will be integrated into the RAIDO Toolbox, delivering optimized, scalable, and energy-efficient solutions for a wide, diverse range of downstream tasks. Collectively, they enable the AI-Model Optimization Layer to provide energy-efficient training solutions, supporting knowledge distillation, as well as few- and zero-shot learning. This comprehensive approach ensures that the component reduces the energy consumption for AI models deployed by the RAIDO platform's pilots.

3.4.2 Functional description

The following Table 8 provides a clear, structured summary of the functionality and characteristics of the Energy and Compute Efficient Learning component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*,

outputs and post-conditions, any *API* used to interact with other components, and the *technology* used to implement it.

Table 8. Energy and Compute Efficient Learning component: Functional description.

Component ID	C04
Component name	Few- & Zero-Shot adaptation and distillation of Vision-Language Models
Sub-name	Energy and Compute Efficient Learning
Description	The core functionality of this component is to enable energy- and data-efficient training of Vision-Language (V-L) models and time-series models using knowledge/model distillation techniques, as well as adaptation methods. The component processes input data, including V-L and time-series data, and routes it through specialized pods for knowledge distillation and/or soft prompt learning, depending on task requirements.
Inputs & pre-conditions	i. Images ii. Time-series
Outputs & post-conditions	Optimized AI model
API	Under development
Implementation technology	Programming language: Python Frameworks/Libraries: Docker, Pytorch

3.4.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 5.

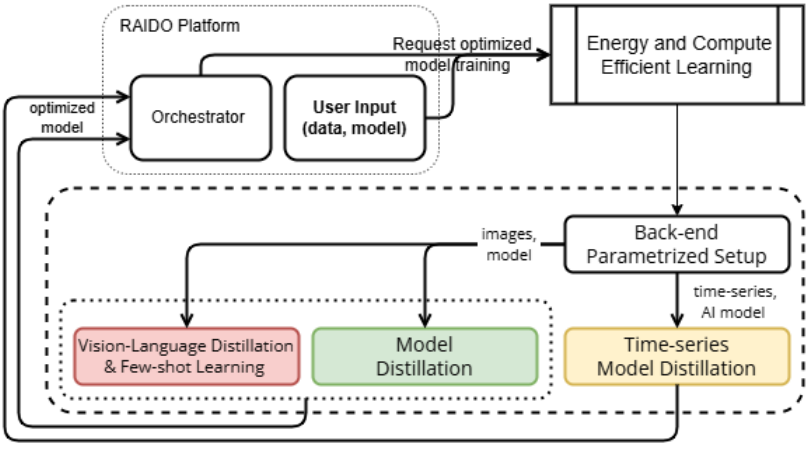

Figure 5. Energy and Compute Efficient Learning: Internal architecture.

3.4.4 Interaction with other components

The Energy & Compute Efficient Learning component interacts with the components listed in Table 9 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 9. Energy and Compute Efficient Learning component: Interactions.

Interacting Component	Description

RAIDO Orchestrator	The component serves as a responsive module within RAIDO, receiving optimized training requests from the Orchestrator to address specific tasks and needs, such as model distillation and few-/zero-shot learning for image or time-series data. Upon receiving these requests and the specified preferred method, it selects the appropriate subsystem to efficiently train the provided model. This interaction ensures the delivery of efficient training methodologies back to the Orchestrator, supporting downstream pipelines and enhancing RAIDO's ability to produce energy-efficient models for diverse use cases.

3.5 Life-Long Learning in Federated Architectures

3.5.1 Business logic

The business logic for this component focuses on developing the second (2nd) part of the AI-Model Optimization Layer, focusing on Lifelong Learning within a Federated Architecture. Federated Continual Learning (FCL) combines Federated Learning and Continual Learning to enable privacy-preserving model training among clients who own dynamic data. Federated Learning decentralizes model training across multiple client devices, avoiding the need to transmit raw data. Continual Learning incrementally updates models as new data becomes available, mitigating catastrophic forgetting and enabling models to retain previously acquired knowledge during retraining.

This Federated & Continual Learning component will utilize the Flower framework to manage federated learning tasks in distributed architectures. Flower provides a collection of federated learning baselines offering solutions that can be used within RAIDO's orchestrator, supporting privacy-preserving federated learning. In addition, the component will utilize the Avalanche framework, an open-source library for continual learning, offering modular solutions for both class- and instance-incremental learning. With the support of Avalanche, this component will allow model designers to select from various incremental learning methods, effectively mitigating catastrophic forgetting during training.

To enhance its capabilities, the Federated & Continual Learning component will incorporate additional continual learning methods focused on preventing overfitting. For example, Learning Without Forgetting (LwF) proposes to adapt a vision model to a new task while preserving performance on original tasks, even without access to the original tasks' training data. Such methods enhance model robustness, especially in scenarios where RAIDO's pilots have limited data availability. In addition, the component will address concept drift by embedding adaptive mechanisms that recalibrate models in response to shifts in data distribution, ensuring stable and reliable model performance over time.

Finally, online meta-learning algorithms will be employed to support dynamic model adaptation. For instance, a small parametric module will be attached to a target network and will learn to predict target losses for unlabeled inputs. Then, this module can propose data points where the target model is likely to make incorrect predictions, facilitating task-agnostic learning. Such methods allow networks to generalize effectively from a single loss function, irrespective of specific downstream tasks.

Overall, the Federated & Continual Learning component comprises several online learning, meta-learning and federated learning technologies, supported by Python and PyTorch libraries. Key frameworks such as Flower and Avalanche will serve as the foundation for building scalable, privacy-conscious, and efficient learning models.

3.5.2 Functional description

The following Table 10 provides a clear, structured summary of the functionality and characteristics of the Federated and Continual Learning component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 10. Federated & Continual Learning component: Functional description.

Component ID	C05
Component name	Life-Long Learning in Federated Architectures
Sub-name	Federated & Continual Learning
Description	The objective is to develop the second part (2nd) of the AI-Model Optimization Layer, focusing on Lifelong Learning within a Federated Architecture. This involves creating new global distillation methods, learning strategies to prevent overfitting, and implementing confidence-based sampling methods to leverage unlabeled external data. In distributed architectures, a Federated Continual Learning (FCL) approach based on distillation will be implemented to minimize data storage needs and requirements and avoid retraining from scratch, while ensuring protection from catastrophic forgetting.
Inputs & pre-conditions	Data and AI models as inputs; data will be required to be in a specific format for training the AI models
Outputs & post-conditions	Retrained AI models as outcome
API	Under development
Implementation technology	Programming Language: Python Frameworks/Libraries: Pytorch, Avalanche (Continual Learning), Flower Framework (Federated Learning)

3.5.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 6.

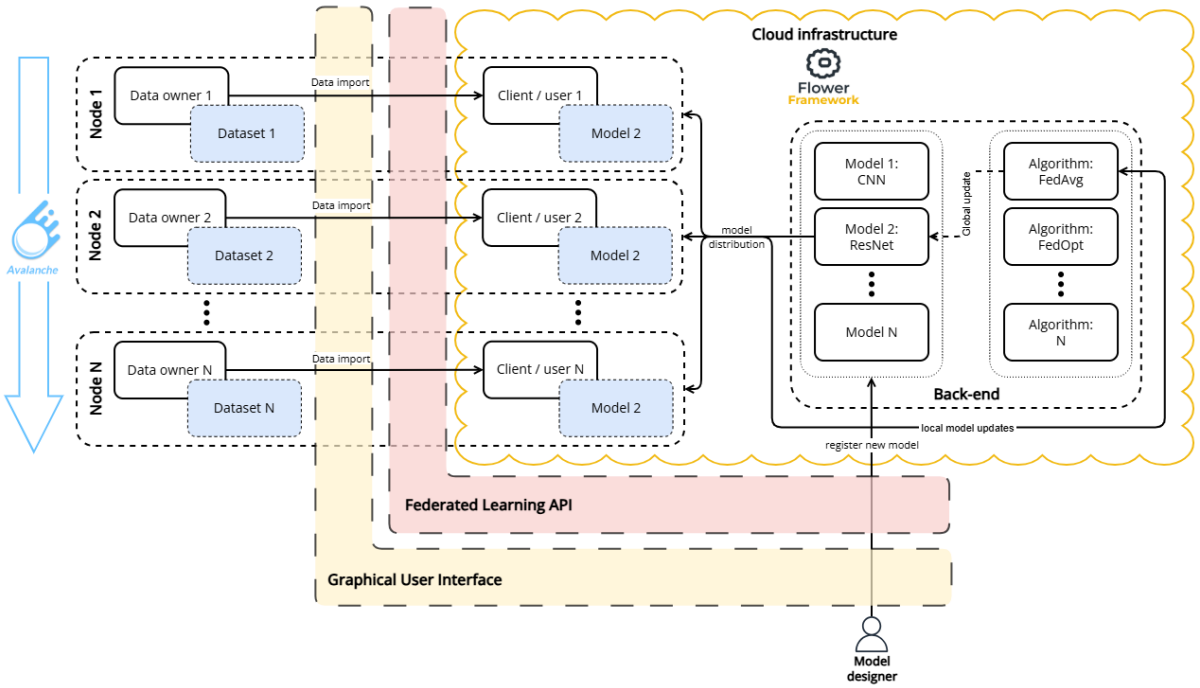

Figure 6. Federated & Continual Learning component: Internal architecture.

3.5.4 Interaction with other components

The Federated & Continual Learning component interacts with the components listed in Table 11 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 11. Federated & Continual Learning component: Interactions.

Interacting Component	Description
RAIDO Orchestrator	The component is part of RAIDO's platform back-end, so it should communicate with the Orchestrator to handle external calls to the dashboard, both to retrieve data from the user and to return the outcome.

3.6 Data Lake

3.6.1 Business logic

T5.3 focuses on establishing the RAIDO Information Layer by implementing advanced Data Lakes to support efficient AI model training and deployment. The primary objective is to optimize the training process, while reducing both data and energy consumption. This task involves leveraging techniques for data ingestion, storage, security, analytics, and governance to create a robust infrastructure for AI development.

The proposed Data Lake will store AI models and datasets while utilizing advanced AI analytics to streamline the identification of optimal ML algorithms and hyperparameters.

The system will explore fine-grained search spaces, including optimization methods, initialization settings, and model configurations. This process is designed to enhance efficiency and scalability in AI training while minimizing computational overhead.

The data lake will serve as a centralized repository for storing vast amounts of raw, structured, semi-structured, and unstructured data at any scale. Unlike traditional databases, a data lake stores data in its native format until it is needed, providing flexibility and scalability for analytics and processing. The data lake will integrate four (4) distinct databases, namely MinIO, MySQL, InfluxDB, and MongoDB. These databases will handle diverse data modalities. MinIO will store the optimized AI models, MySQL will retain structured relational data, InfluxDB is optimized for handling time-series data, and MongoDB will store unstructured and/or semi-structured data.

Delving into these databases:

- MinIO is a high-performance, distributed object storage system designed for handling large-scale unstructured data. MinIO offers features such as high availability, data redundancy, and strong security through encryption and access control, ensuring that data is both protected and readily accessible. Its lightweight and open-source nature make it a flexible solution for organizations that need a cost-effective yet powerful storage solution.
- InfluxDB is an open-source time-series database designed for high-speed data ingestion and retrieval, specifically for metrics, events, and analytics. Optimized for time-stamped data, it is ideal for use cases such as monitoring systems, Internet of Things (IoT) applications, and real-time analytics. Its scalability and ability to handle high write and query loads make it a preferred choice for managing time-sensitive data in modern, performance-driven applications.
- MySQL is a popular open-source relational database management system (RDBMS) that employs Structured Query Language (SQL) to manage and organize data. Known for its speed, reliability, and flexibility, MySQL is widely used in web applications, enterprise systems, and data-driven services. It supports complex queries, transactional processing, and multi-user access, making it suitable for a wide range of use cases.
- MongoDB is a leading NoSQL database designed to store and manage unstructured or semi-structured data in a flexible, document-oriented format. Unlike traditional relational databases, MongoDB uses collections of JSON-like documents, which makes it well-suited for dynamic schemas and applications requiring scalability and high performance. It supports features like horizontal scaling, replication for high availability, and indexing for faster query processing.

3.6.2 Functional description

The following Table 12 provides a clear, structured summary of the functionality and characteristics of the Data Lake component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 12. Data Lake: Functional description.

Component ID	C06
Component name	Data Lake
Sub-name	Data Lake
Description	The task is dedicated to developing the RAIDO Information Layer using Data Lakes to improve the efficiency of AI model training and deployment. The focus is on implementing robust data ingestion, storage, security, analytics, and governance frameworks. By utilizing advanced AI analytics, the task will optimize ML processes by exploring model configurations, initialization methods, and hyperparameter tuning, thereby reducing energy and data requirements during training and deployment. Overall, it aims to establish a scalable, efficient, and sustainable AI ecosystem within RAIDO, setting new standards for energy-efficient and effective data processing in AI workflows.
Inputs & pre-conditions	Data and AI models as inputs; data will be required to belong in one of the supported data modalities (tabular, images, text, time-series).
Outputs & post-conditions	Data and AI models as outputs that will be further utilized in the subsequent parts of the RAIDO platform.
API	Under development
Implementation technology	Tabular Datasets: MySQL Images/Text: MongoDB Time-series: InfluxDB AI models: MinIO

3.6.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 7.

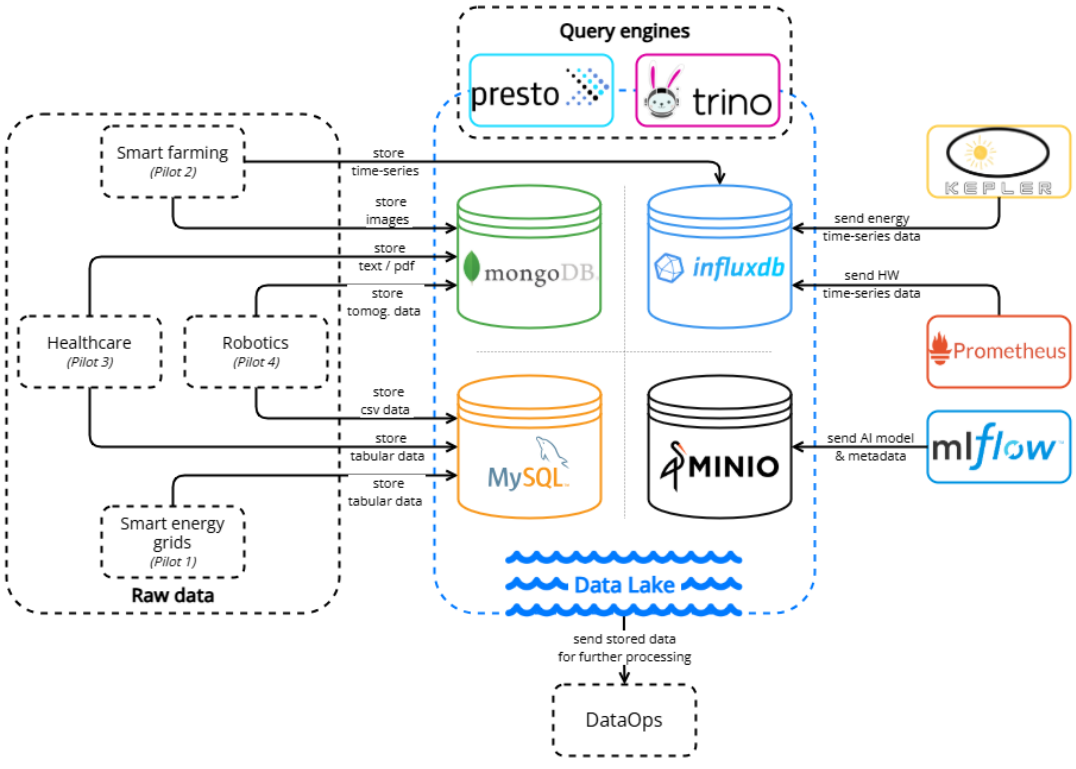

Figure 7. Data Lake component: Internal architecture.

3.6.4 Interaction with other components

The Data Lake component interacts with the components listed in Table 13 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 13. Data Lake component: Interactions.

Interacting Component	Description
Visualization Engine	The GUI provides an intuitive interface for users to interact with the RAIDO platform. It allows users to input various data formats (tabular, images, text, time-series), configure AI model parameters, and manage training and deployment tasks. The GUI integrates seamlessly with backend components, enabling smooth data flow and providing real-time feedback, visualizations, and alerts. It simplifies complex AI processes, making it easy for users to optimize models and monitor performance without requiring deep technical knowledge.

3.7 Frameworks for AI explainability

3.7.1 Business logic

This component aligns with the overall RAIDO objective to develop AI models, data, and solutions that meet the requirements of trustworthy AI, emphasizing accuracy, robustness, safety, ethical principles, and reliability, in line with the European Approach to AI. The eXplainable AI (XAI) models and services, along with the bias and ethics evaluation processes developed in this task, will form part of the Trustworthy AI Layer. This layer will autonomously search and evaluate processes for safety, robustness, fairness, privacy, and reusability. Specifically, it will simultaneously search a fine-grained space to identify the appropriate model, ensure robustness, meet energy/cost requirements, and enhance reliability and accuracy, thus enabling trustworthy AI solutions. The entities within the Trustworthy AI Layer will be accessible to all AI and data processing components of the RAIDO solution.

Regarding explainability, despite some advancements in XAI techniques, humans still struggle to understand the detailed descriptions of an AI model's inner workings and the often imprecise post-hoc explanations [6] , [7] , [8] , [9] , [10] , [11] . In an effort to improve explainability in “black-box” AI systems (i.e. increase interpretability and transparency and improve their ability to reason & rationalize fairly), the RAIDO XAI engine aims to develop techniques that improve the understanding of how a model arrives at a particular decision. As part of the Trustworthy AI Layer, this engine will directly or indirectly connect with all other components of the platform.

The XAI engine will use a game-theoretic method based on SHapley Additive Explanations (SHAP) [12] to explain AI models. SHAP estimates the average marginal

contribution of a feature value to provide a global understanding of the significance of each feature. Then, for each AI process, Local Interpretable Model-agnostic Explanations (LIME) schemes [13] , [14] will be created to improve model interpretability. LIME builds sparse linear models based on chosen predictions to explain how the model behaves locally. In addition, the methods of Layer-wise Relevance Propagation (LRP) [15] [16] , counterfactual explanations [17] , and Gradient-weighted Class Activation Mapping (CAM) [18] will also be used to explain AI outputs in the context of their inputs. Furthermore, the envisaged XAI engine will offer solutions and techniques to detect and prevent model or data bias, supporting human-centered and ethical AI model development. Bias and ethics evaluation will be conducted through reverse-engineering latent spaces of pre-trained AI models. All these methods will be supported by lightweight blockchain tools for accountability and traceability, utilizing RAIDO's certification processes.

Within the overall RAIDO architecture, the XAI engine will establish two (2) main checkpoints, as described below. The first checkpoint will occur at the RAIDO input stage, where it will evaluate the data provided by the user. This will be an automated component that assesses the data generation, semantic enrichment, and curation stages of the platform. The second checkpoint will be positioned after platform's optimization stages and will evaluate the training and validation stages of the AI model development, aiming to deliver a trustworthy and optimized solution.

The performance of the XAI engine will be evaluated based on several criteria, including its explainability (rated +1 on a 5-point Likert scale), prediction accuracy (>90%), model explainability with SHAP and LIME (measured by Area Under the Curve, >85%), reduction of defective or faulty outcomes and products (>12%), and increased acceptance of Human-in-the-Loop (HITL) AI solutions by society (+2 in a 5-point Likert scale).

3.7.2 Functional description

The following Tables 14, 15 provide a clear, structured summary of the functionality and characteristics of the XAI component (divided into two (2) checkpoints). They outline the components' *ID*, both their *actual* and *sub-names*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technologies* used to implement them.

Table 14. XAI component (a): Functional description.

Component ID	C07a
Component name	XAI Engine Checkpoint #1
Sub-name	XAI
Description	This component evaluates the data generation, semantic enrichment, and curation stages of the platform. It will operate through two trust levels (TrLs): • TrL1: The reliability and transparency of the data sources will be explicitly examined to identify data samples that do not

	satisfy the required quality standards. In addition, the data collection and annotation techniques will be assessed to ensure compliance with relevant bias criteria in alignment with the EU legislation. • TrL2: Subsequently, the feature pre-processing and data labelling processes will be monitored, including outlier detection, data balancing, curation, and novelty discovery. During this stage, equality, ethics, privacy, and uncertainty awareness will be prioritized and ensured.
Inputs & pre-conditions	Input data
Outputs & post-conditions	XAI-approved data
API	Under development
Implementation technology	Programming Language: Python Frameworks/Libraries: fairlearn, pandas, cleanlab, provenance

Table 15. XAI component (b): Functional description.

Component ID	C07b
Component name	XAI Engine Checkpoint #2
Sub-name	XAI
Description	This component evaluates the training and validation stages of the AI model development to deliver a trustworthy and optimized solution. It will operate through two (2) more TrLs: • TrL3: The learning methods and algorithms will be assessed for semantics for their familiarity (i.e. understanding of the specific domain or context), knowledgeability (i.e. access to and utilization of accurate and complete knowledge relevant to the task), and fairness (i.e. ensuring decisions are made without prejudice or discrimination against certain groups). Note that these attributes are all domain specific. • TrL4: Subsequently, the aspects of performance, model variance or bias, and in situ comparison will be investigated. Methods based on reverse-engineering the latent space of the trained models will be employed to provide a model-agnostic solution for bias detection, ensuring key qualities such as consistency and local accuracy.
Inputs & pre-conditions	AI models
Outputs & post-conditions	XAI performance metrics
API	Under development
Implementation technology	Programming Language: Python Frameworks/Libraries: SHAP, EBM, LIME, ELI5, Dalex, and Shapash

3.7.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 8.

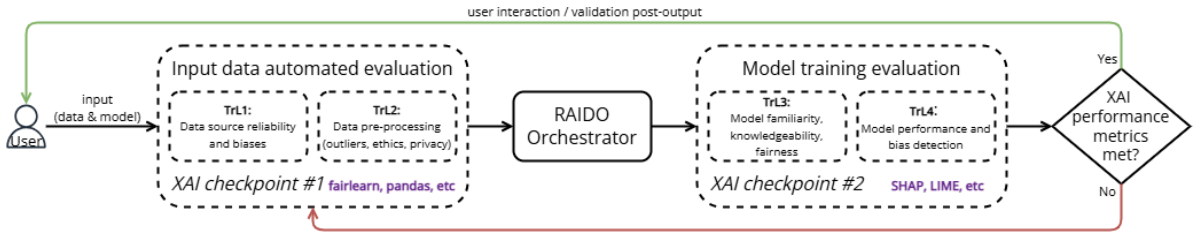

Figure 8. XAI component: Internal architecture.

3.7.4 Interaction with other components

The XAI component interacts with the components listed in Table 16 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 16. XAI component: Interactions.

Interacting Component	Description
Data Enrichment	Prior to any optimization procedure within RAIDO, the XAI Engine Checkpoint #1 evaluates the source and pre-processing of the input data.
AI optimization (RAIDO orchestrator)	The XAI-approved data from XAI Engine Checkpoint #1 are forwarded to the Orchestrator, which handles the core RAIDO functionalities. After optimization, the resulting models are evaluated in XAI Engine Checkpoint #2 for their learning methods, performance, and biases. In case the XAI performance metrics are satisfied, the Orchestrator's output is forwarded to the user.

3.8 Reinforced benchmarking and feedback-based progress monitoring

3.8.1 Business logic

This component develops the feedback loop within the RAIDO architecture, enabling an iterative optimization process for services and processes using reinforcement learning within a dynamic and autonomous evaluation pipeline. Specifically, RAIDO utilizes multi-objective reinforced active learning [19] , where deep reinforcement learning agents are optimized and fine-tuned according to various preferred characteristics, such as energy efficiency or explainability. To enhance transparency in the training process and avoid biases, RAIDO adopts a feedback-based architecture that integrates inputs from humans or human-centric sources (HITL) and extracted analytics to improve learning outcomes [20] , [21] . The primary goal of this architecture is to leverage feedback-based learning techniques to optimize AI model efficiency while meeting the defined policy or optimization expectations.

In addition, we explore the implementation of simple lower-level self-optimizations (e.g., at feature level) to support higher-level optimization across the application. One significant advantage of this approach is its scalability; as the RAIDO framework

increases in complexity, distributing optimization workloads to lower levels helps maintain scalability. This approach can also lead to emergent behavior, where the interactions between different (optimized) features yield unexpected capabilities beyond their explicit programming (i.e. an AI system that is greater than just the sum of its parts) [22] .

Benchmarking within the evaluation pipeline will rely on multi-objective Key Performance Indicators (KPIs), automatically selected to accurately measure model performance across various domains and applications. At this stage, it is essential to incorporate fairness and diversity to ensure that the benchmarking system is generalizable and applicable across different models, domains, and datasets. KPIs will be autonomously selected using ML and deep learning techniques [23] or genetic algorithms [24] .

3.8.2 Functional description

The following Table 17 provides a clear, structured summary of the functionality and characteristics of the Feedback Loop component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 17. Feedback Loop component: Functional description.

Component ID	C08
Component name	RL Benchmarking and Feedback Loop
Sub-name	Feedback Loop
Description	The RAIDO Feedback Loop Module is designed to optimize system performance and ensure compliance with predefined Service Level Objectives (SLOs) by integrating sophisticated data processing and reinforced learning techniques. The module receives complex operational parameters or data, which are initially simplified into a manageable, encoded format to facilitate efficient analysis. This streamlining capability is crucial for enabling the system to process inputs effectively and generate optimized, actionable outputs. Using reinforced learning and multi-objective optimization techniques, the module dynamically adjusts these parameters to improve system operations and meet specified performance standards. The optimized outputs are then converted into practical parameters, ready for implementation, continually refining the system's performance. In addition, the module incorporates a HITL mechanism to oversee automated decisions, ensuring alignment with operational realities and offering an essential check against potential errors or misalignments that may arise from fully automated outputs.
Inputs & pre-conditions	• Operational data or model parameters from the Orchestrator in a simplified (encoded) format. • Defined SLOs and performance metrics.
Outputs & post-conditions	• Compliance reports comparing current performance against SLOs. • Notifications and alerts sent to the HITL for significant events or decisions.

	• Optimized parameters sent back to the Orchestrator for subsequent runs.
API	Under development
Implementation technology	Programming Language: Python | TensorFlow, PyTorch Frameworks/Libraries: FastAPI, Docker

3.8.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 9.

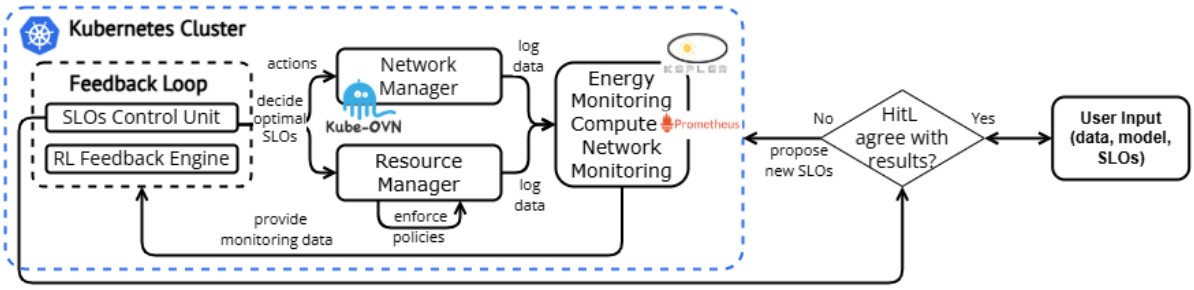

Figure 9. Feedback Loop component: Internal architecture.

3.8.4 Interaction with other components

The Feedback Loop component interacts with the components listed in Table 18 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 18. Feedback Loop component: Interactions.

Interacting Component	Description
AI optimization (RAIDO Orchestrator)	The RL Loop module receives as inputs a set of actions performed by the AI optimization module (actions), along with KPIs and platform results (data or model parameters). The feedback loop proposes to the Orchestrator a set of modified actions that needs to be rerun to achieve further optimization.
HITL	The HITL monitors the outputs and decisions generated by the RL Loop module to ensure alignment with operational standards and expectations. This provides a crucial layer of human oversight. In addition, the HITL can provide direct feedback to influence adjustments in operational tactics or SLO configurations, ensuring system integrity, adaptability, and compliance.

3.9 Adaptive AI processes and visualization engine

3.9.1 Business logic

The RAIDO Adaptive Visualization Engine is designed to deliver dynamic, user-centric visualizations that adapt to the specific needs and contexts of end users. This engine integrates HITL support and autonomous reporting capabilities to enhance data creation, processing, and AI model optimization within the RAIDO platform.

HITL support allows users to provide feedback during the visualization process, enhancing both the quality and usability of the outputs. Users can interact with visualizations, and their input directly influences the behavior of the AI models. Meanwhile, autonomous reporting automatically generates tailored reports that summarize key insights and findings from the visualizations, ensuring these reports meet the specific needs and preferences of different user groups.

The business logic of this engine includes well-defined rules and conditions: (i) User interaction rules, where visualizations must update in real-time based on user interactions and inputs. In addition, user feedback should be incorporated into the AI model optimization process. (ii) Data handling rules that ensure data privacy and security must be maintained throughout all visualization stages. Robust validation and error-handling mechanisms ensure data integrity at every step.

Performance considerations include optimal system efficiency and user experience by optimizing data processing and visualization algorithms to ensure real-time performance and implementing caching and other latency-reduction techniques to improve user responsiveness. This business logic framework ensures that the RAIDO Adaptive Visualization Engine delivers high-quality, user-centric visualizations and reports, significantly enhancing the overall efficiency and effectiveness of the RAIDO platform.

3.9.2 Functional description

The following Table 19 provides a clear, structured summary of the functionality and characteristics of the Visualization Engine component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 19. Visualization Engine component: Functional description.

Component ID	C09
Component name	Adaptive AI processes and visualization engine
Sub-name	Visualization Engine
Description	The objective is to develop a dynamic visualization framework that serves as the foundation for RAIDO's data creation, processing, and AI model optimization solutions. This framework is designed to meet end-user needs by delivering tailored, multi-dimensional visualizations that incorporate domain-specific knowledge and adaptable contexts for diverse use cases. The AI-powered visualization engine will provide optimal data, model, and XAI visualizations aligned with user requirements, leveraging advanced data analytics and autonomous reporting tools. It will serve as the central interface for user interactions across all stages, processes, and components of the RAIDO platform. By integrating user feedback, system requirements, and HITL interfaces, the engine will facilitate efficient and effective data generation and AI optimization. In addition, it will support digital twins (DTs) and adaptive visualization

	tools to enhance the overall functionality, usability, and performance of the RAIDO platform.
Inputs & pre-conditions	User inputs
Outputs & post-conditions	Visualizations
API	Under development
Implementation technology	UI Frameworks: Bootstrap, Vue.js HTTP Client: Axios Dashboard back-end: Python, Node.js Chart Frameworks: D3, Chart.js Sockets Framework: Socket.io, ngx-socket-io

3.9.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 10.

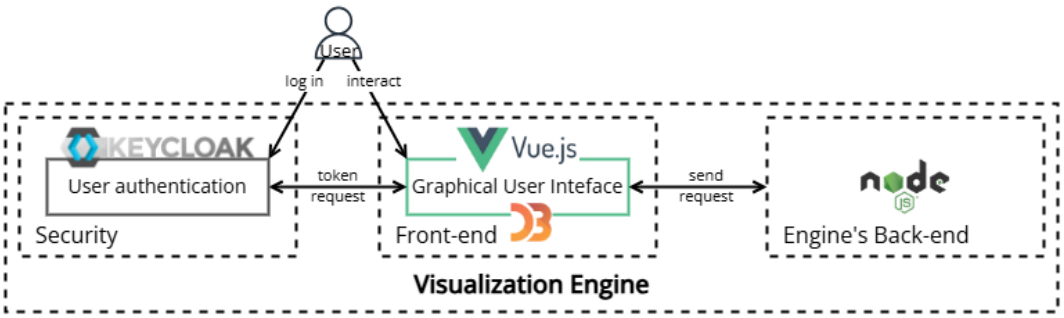

Figure 10. Visualization Engine component: Internal architecture.

3.9.4 Interaction with other components

The Visualization Engine component interacts with the components listed in Table 20 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 20. Visualization Engine component: Interactions.

Interacting Component	Description
RAIDO Orchestrator	The Visualization engine gets the required data by querying directly the RAIDO's back end that is being handled by the Orchestrator.

3.10 Self-evolving green-AI and data orchestration

3.10.1 Business logic

The Green AI and Data Orchestrator (part of RAIDO Orchestrator) is a core component of the RAIDO project, designed to manage AI and data workflows while minimizing computational waste and optimizing energy usage. This enables RAIDO to deliver robust AI capabilities with a reduced environmental impact from the associated computationally intensive operations. Its primary role is to provide optimal data and model configurations that align with user-defined requirements for performance, energy

usage, and computational resources, thereby establishing efficient AI workflows within specified sustainability thresholds.

Key functionalities of the Green AI and Data Orchestrator are the following:

- Workflow Orchestration: Manages data flows and task scheduling, optimizing AI and data pipelines to maximize energy efficiency and model performance.
- AI Model Lifecycle Management: Supports experiment tracking and the deployment of AI models.
- API Integration for Adaptive Visualization Engine (AVE) and System Connectivity: Establishes robust APIs to ensure smooth communication between the user interface (UI) and other components within the RAIDO architecture.
- HITL support: Incorporates end-user input and feedback, allowing them to import data and models, define energy and performance constraints, and manage workflow configurations.
- Autonomous Decision-Making: Automatically selects optimal data, pre-processing, and model training methods based on specified energy and performance requirements.
- Intelligent Data Augmentation: Utilizes synthetic data and generative AI techniques to improve model performance and ensure an optimal fit to the data and application domains specified by the user.

The Green AI and Data Orchestrator realizes its functionalities through a well-integrated architecture that leverages API-based connectivity for seamless communication with other RAIDO components. This setup enables real-time responsiveness and precise user control, integrating workflow updates and data exchanges with RAIDO's UI. In practice, the Orchestrator receives input and feedback from end-users via RAIDO's Adaptive Visualization Engine (AVE). Users import their models, define data and application domains, and set performance and energy constraints. The AVE compiles this information into a JSON file, which is then forwarded to the Orchestrator's API. This API also facilitates communication with other essential components crucial to the Orchestrator's functionality, such as Data Lakes, Model Pools, and the edge-to-cloud (E2C) Orchestrator. Upon receiving the JSON, the Orchestrator autonomously selects and iteratively executes the most suitable data and AI pipeline based on the specifications provided. This iterative process continues until the user-defined requirements are satisfied or until specified thresholds are met or exceeded. This adaptive, self-evolving architecture continuously optimizes processes, ensuring that RAIDO's AI and data workflows remain efficient, sustainable, and aligned with the project's energy-conscious objectives.

3.10.2 Functional description

The following Table 21 provides a clear, structured summary of the functionality and characteristics of the Green-AI component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 21. Green-AI component: Functional description.

Component ID	C10
Component name	Green AI and Data Orchestrator
Sub-name	Green-AI
Description	The core functionality of the Green AI and Data Orchestrator is to manage AI and data workflows, optimizing energy usage and minimizing computational waste. It receives input from end-users via RAIDO's dedicated UI, including model specifications and performance constraints, and autonomously configures AI and data pipelines to meet these requirements. The Orchestrator continuously adjusts workflows to ensure efficiency, sustainability, and alignment with predefined energy and resource thresholds.
Inputs & pre-conditions	End-user input and HITL feedback
Outputs & post-conditions	• Optimized Data and AI pipelines • AI model performance metrics • Energy efficiency metrics
API	Under development
Implementation technology	Programming Language: Python Frameworks/Libraries: Airflow, MLflow, FastAPI, (Tensorflow, Keras, Pytorch, Scikit-learn)

3.10.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 11.

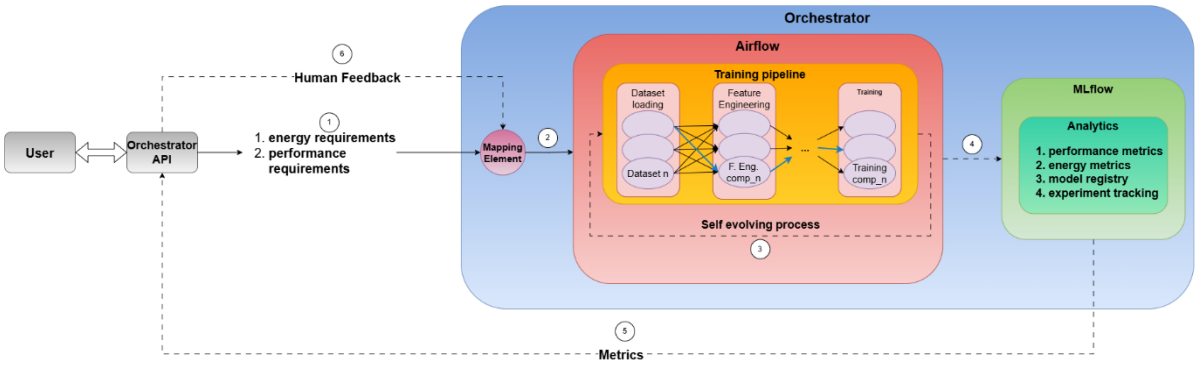

Figure 11. Green-AI component: Internal architecture.

3.10.4 Interaction with other components

The Green-AI component interacts with the components listed in Table 22 below. This table provides an overview of each *interacting component* and a *description* of their

respective interactions, highlighting how data or functionality is exchanged between them.

Table 22. Green-AI component: interactions.

Interacting Component	Description
Data Lake	Downloads and pushes the appropriate data and datasets from the Data Lake based on the data and application domain selected by the end-user. Imports models that best fit the selected data and the user-defined requirements.
Visualization Engine	Enables communication with end-users through the RAIDO dashboard by triggering the Orchestrator's optimization process once the user's specifications are completed in the dashboard.
HITL feedback	Reinforcement learning-based feedback and monitoring mechanisms (pending). Communication with HITL feedback component is under investigation. To be finalized soon.

3.11 Networking management

3.11.1 Business logic

The role of the Network Manager (part of RAIDO Orchestrator) is to secure the onboarding and operation of a consistent network overlay among edge and cloud nodes and to facilitate the selection of a cluster-head that will represent an entire physical deployment. The purpose of the network overlay is to:

- Generate and store network metrics to then feed an AI model for Service Level Objectives (SLOs) adaptations.
- Manage virtual ports, IP addresses, and pods communication to establish a virtual overlay.
- Manage virtual overlay networks across multiple pods.
- Participate in the selection process of a cluster-representative (cluster-head) to which several computational tasks can be offloaded.

Semi-centralized techniques will be employed for cluster-node selection. Each node is not statically configured with the address of a super-node. Instead, prior to the joining protocol, the node executes a cluster-head election protocol. The objective of such protocols is to logically split the network topology into distinct "islands", each represented by a cluster-head. The cluster-head election is performed through a combination of controlled flooding schemes, initiated randomly by volunteer nodes based on their capabilities and available resources (e.g., battery, memory, etc.). Nodes within a certain proximity that receive these broadcasts and do not have a super-node will accept the broadcasting node as their cluster-head (as super-node). Consequently, during network joining or leaving, nodes retrieve their "topological position" from the elected node.

3.11.2 Functional description

The following Table 23 provides a clear, structured summary of the functionality and characteristics of the Network Manager component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 23. Network Manager component: Functional description.

Component ID	C11a
Component name	Networking management
Sub-name	Network Manager
Description	The core functionality of this component is to manage virtual overlay networks between edge and cloud environments. Its input consists of network rules that dynamically update routing or replace a pod within the cluster at runtime. The component continuously feeds itself with new metrics to optimize the deployments from a network perspective.
Inputs & pre-conditions	Network metrics
Outputs & post-conditions	Network SDN rules
API	Under development
Implementation technology	Programming Language: Python Frameworks/Libraries: Kube-OVN, Flask

3.11.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 12.

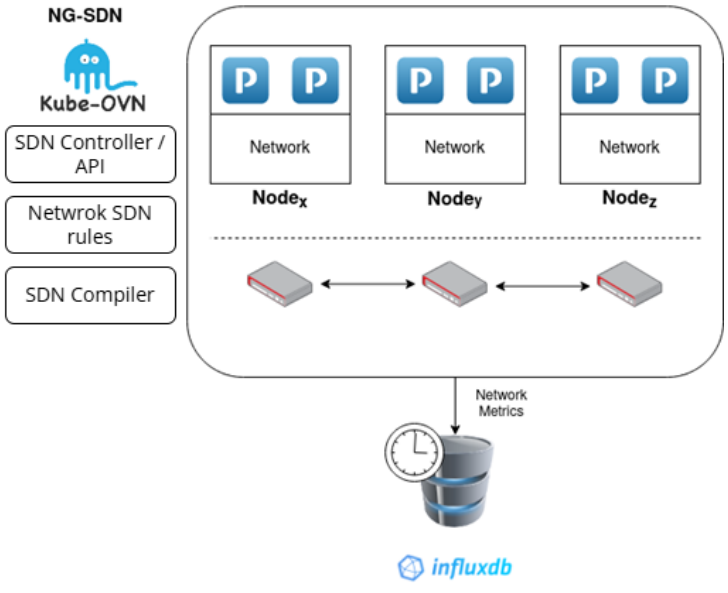

Figure 12. Network Manager component: Internal architecture.

3.11.4 Interaction with other components

The Network Manager component interacts with the components listed in Table 24 below. This table provides an overview of each *interacting component* and a *description*

of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 24. Network Manager component: Interactions.

Interacting Component	Description
Data Lake	It pushes aggregate network metrics to the logically centralized measurement collection component.
Visualization Engine	It interacts with the RAIDO dashboard of the Visualization Engine component to report back the network status of the E2C infrastructure.

3.12 AI-based E2C continuum resource allocation & deployment

3.12.1 Business logic

The Resource Manager (part of RAIDO Orchestrator) is a key component in RAIDO, within a Kubernetes cluster, responsible for orchestrating task execution across edge and cloud resources. It determines optimal task placement and scheduling by formulating and solving optimization models tailored to specific use cases, ensuring compliance with defined SLOs. These models consider metrics such as energy consumption, latency, bandwidth, and resource availability, leveraging optimization techniques and AI-driven approaches to achieve efficient SLO decision-making with reduced runtime. The Resource Manager dynamically adapts to workload demands through both horizontal scaling, by adjusting the number of servers, and vertical scaling, by optimizing resources such as CPU and RAM for individual servers. It enforces smart policies derived from real-time monitoring of energy consumption and performance metrics, refined through AI-/ML-based optimization. For example, during high RAM utilization, it can allocate additional memory to maintain performance. By integrating monitoring data with its optimization processes, the Resource Manager ensures tasks are scheduled and executed efficiently, balancing energy efficiency, resource utilization, and latency to meet the defined SLOs.

3.12.2 Functional description

The following Table 25 provides a clear, structured summary of the functionality and characteristics of the Resource Manager component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 25. Resource Manager component: Functional description.

Component ID	C11b
Component name	AI-based E2C continuum resource allocation & deployment
Sub-name	Resource Manager
Description	Based on the specific SLOs associated with use case and the availability of the resources, the Resource Manager determines the

	appropriate layer for task execution, i.e. edge, cloud, and how tasks will be scheduled within each layer after the offloading decisions have been made.
Inputs & pre-conditions	SLOs of use cases;
Outputs & post-conditions	Availability of resources
API	Under development
Implementation technology	Under development

3.12.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 13.

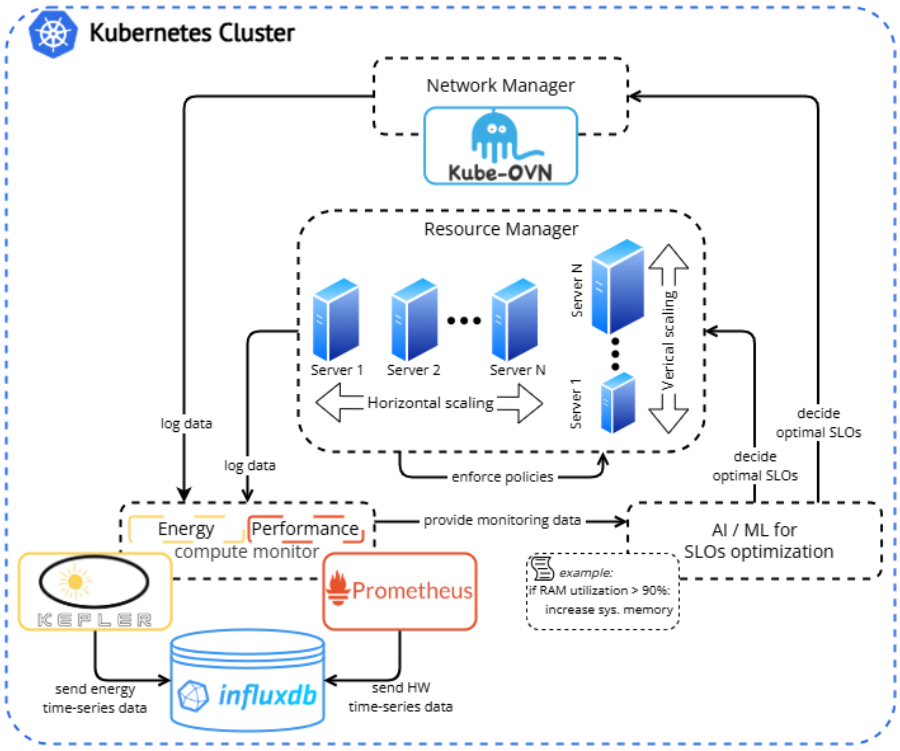

Figure 13. Resource Manager component: Internal architecture.

3.12.4 Interaction with other components

The Resource Manager component interacts with the components listed in Table 26 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 26. Resource Manager component: Interactions.

Interacting component	Description
Data lake	The resource manager module of RAIDO will receive input from the data lake regarding compute and network metrics. In turn, the resource manager will also provide input/feedback to the offloading manager component regarding the nodes that successfully executed the tasks, allowing this information to be leveraged for updating the pricing models.
Visualization Engine	The Resource Manager interacts with the UI by providing real-time data on resource availability, task status, and performance metrics (e.g., energy

	consumption, latency, resource utilization). The UI displays this information in an intuitive format, allowing users to monitor and manage resource allocation, offloading decisions, and task scheduling. The Resource Manager also communicates task execution outcomes, scheduling adjustments, and any changes in the system's status, enabling users to visualize the impact of resource management decisions. Additionally, the UI may allow the user to input or modify SLOs or preferences, which the Resource Manager then incorporates into task scheduling and placement decisions.
Network Manager	It interacts with the Network Manager to obtain real-time data on network conditions, such as available bandwidth, latency, and the reliability of the communication links between mobile, edge, and cloud resources. Based on this information, the Resource Manager adjusts its task placement and scheduling decisions, ensuring that tasks are allocated to the most appropriate resource layer (mobile, edge, or cloud) according to network constraints and performance goals. The Resource Manager may also provide feedback to the Network Manager regarding resource consumption, allowing it to optimize network resources further for efficient task execution and offloading.

3.13 Blockchain & Smart Contracts with IPFS Support

3.13.1 Business logic

The RAIDO platform's Blockchain & Smart Contracts component integrates IPFS with Hyperledger Fabric to create a decentralized ledger framework that ensures accountability, transparency, and traceability for AI model optimization and certification. This configuration allows RAIDO to utilize blockchain-based certification to guarantee data integrity, regulatory compliance, and secure model version tracking. The system accomplishes dual-layer validation by incorporating both Regulatory Orgs and the RAIDO Regulator Organization. This process ensures that transaction validations satisfy RAIDO's internal governance requirements, as well as external regulatory norms and standards, establishing a robust certification trail for all platform interactions and optimizations.

3.13.2 Functional description

The following Table 27 provides a clear, structured summary of the functionality and characteristics of the Blockchain component. It outlines the component's *ID*, both its *actual* and *sub-name*, *description*, *inputs and pre-conditions*, *outputs and post-conditions*, any *API* used to interact with other components, and the *technology* used to implement it.

Table 27. Blockchain component: Functional description.

Component ID	C12
Component name	Blockchain & Smart Contracts with IPFS Support
Sub-name	Blockchain
Description	This component certifies data and model updates by executing smart contracts across the Regulatory Organizations and the RAIDO

	Regulator Organization, ensuring compliance with both internal and external standards. Certification records are stored on IPFS for decentralized, immutable storage, with blockchain transactions referencing these records to enable traceable certification.
Inputs & pre-conditions	i. Model data from the AI training pipeline ii. Predefined certification standards iii. Compliance criteria for smart contracts.
Outputs & post-conditions	• Certified transaction records • IPFS-stored data references accessible to stakeholders for traceability.
API	Under development
Implementation technology	Under development

3.13.3 Information flow and internal architecture

The high-level architecture of this component is shown in Figure 14.

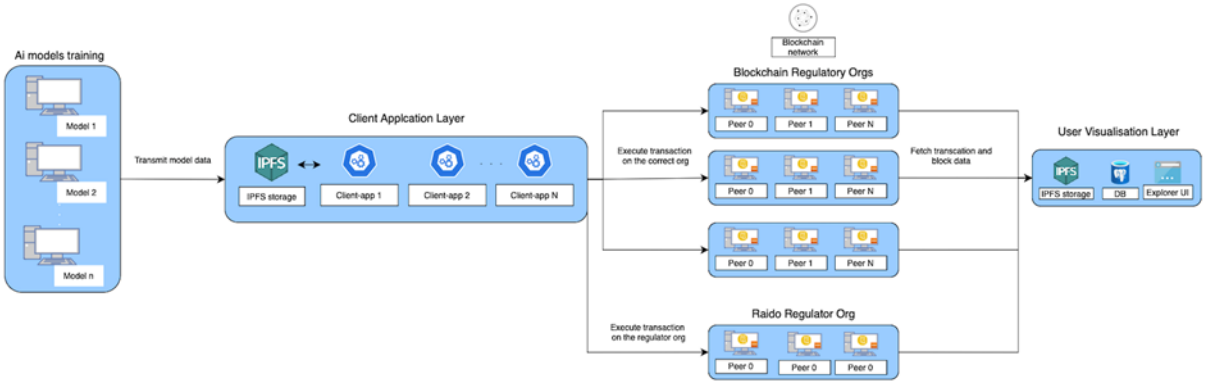

Figure 14. Blockchain component: Internal architecture.

3.13.4 Interaction with other components

The Blockchain component interacts with the components listed in Table 28 below. This table provides an overview of each *interacting component* and a *description* of their respective interactions, highlighting how data or functionality is exchanged between them.

Table 28. Blockchain component: Interactions.

Interacting component	Description
AI-Pipelines	Sends model results (e.g., weights, performance metrics) to the Client Application Layer for certification. Initiates the certification process by transmitting data to be stored on IPFS and verified through blockchain transactions.
Data Lake	Stores certified model data and metadata immutably. Blockchain transactions reference this IPFS-stored data, creating a tamper-proof record that is accessible for audits and compliance checks.
Visualization Engine	Displays certification statuses, transaction histories, and IPFS data references through the Explorer UI, enabling stakeholders to monitor certification and compliance.

4 Requirements Traceability Matrix

The Requirements Traceability Matrix (RTM) provided below establishes the connections between Functional Requirements (FRs) and various key RAIDO elements. Specifically, the RTM links the user requirements and system specifications (which are essentially the non-functional requirements (NFRs) of RAIDO), identified and outlined in the previous deliverable D2.1: “Use Case, KPIs, Requirements, Specification, Slices & Technology Enablers Definition Report”, with the RAIDO FRs and technical components identified and presented in this deliverable.

The RTM ensures alignment between the relevant user needs, the RAIDO specifications, and the technical implementations, including the technical roles of actors and RAIDO components. The RTM Table 29 thus includes the *FR ID*, *title*, and *description*, along with *link to StRs* (stakeholder/user requirements), and *link to RAIDO NFRs* (organized by *ID* and *name*). It also details the associated *RAIDO actors* and *link to the corresponding component IDs* for each FR.

We will continuously monitor the RTM and its comprehensive traceability to assess the progress and ensure that all technical components evolve in line with the RAIDO project objectives.

Table 29. Requirements Traceability Matrix.

FR ID	Title	Description	Link to StRs	Link to RAIDO NFRs	RAIDO actors	Link to component id
FR01	Scalable synthetic data generation for AI models	RAIDO must generate synthetic datasets using Digital Twins, GANs family of models, or Diffusion Models to supplement real-world data. This should enable AI models to train effectively in data-scarce environments while reducing energy consumption.	R01, R27	SP1 - Functional completeness, SP3 - Time behavior, SP4 - Resource utilization, SP8 - Availability, SP14 - Adaptability	AE, DE	C03
FR02	Enrichment and fusion of multimodal data	RAIDO must integrate multimodal data into a centralized data lake with MinIO, InfluxDB, MySQL, and MongoDB. This data should be stored and persisted for training AI models, ensuring high-quality and adaptable insights.	R02	SP1 - Functional completeness, SP4 - Resource utilization, SP5 - Interoperability, SP8 - Availability, SP9 - Confidentiality	DP, AE, DE	C02, C06
FR03	Federated and continual learning for scalable data processing	RAIDO must implement federated learning to process high-frequency, distributed data with advanced data ownership and minimal data transfer. Few-shot learning, transfer learning and distillation models should be leveraged to ensure energy-efficient and scalable AI model training.	R03, R12, R13	SP2 - Functional correctness, SP3 - Time behavior, SP4 - Resource utilization, SP9 - Confidentiality, SP10 - Integrity	AE, DE, PA	C05
FR04	End-to-end secure data management	RAIDO must ensure that all data exchanges are secure (e.g., with Keycloak), compliant with GDPR and other international regulations, and support immutability. Blockchain integration should be used to safeguard proprietary data and ensure auditability.	R04	SP8 - Availability, SP9 - Confidentiality, SP10 - Integrity, SP11 - Authenticity, SP13 - Testability	AE, DE, PA	C06, C09, C12

FR ID	Title	Description	Link to StRs	Link to RAIDO NFRs	RAIDO actors	Link to component id
FR05	Adaptive optimization (few-/0-shot, distillation, dimensionality reduction, etc.) for AI model training	RAIDO must dynamically adjust training configurations both on the algorithmic side, as well as on the HW platform to optimize performance and energy consumption. The system should perform adaptive fine-tuning based on feedback of energy consumption, CPU, RAM, and bandwidth utilization to achieve an optimal balance between performance and energy efficiency.	R05, R06, R10, R11, R15, R16, R17, R18, R19, R21, R22, R26, R29, R30, R32, R34, R35	SP3 - Time behavior, SP4 - Resource utilization, SP7 - User engagement, SP8 - Availability, SP14 - Adaptability	AE, DE, PA	C08, C10, C11a, C11b
FR06	Compliance with regulatory and legal frameworks	RAIDO must comply with EU regulations (e.g., GDPR, MDR) and trustworthy AI standards. The system should provide tools for compliance tracking, auditing, and generating reports for stakeholders across healthcare and energy domains.	R07, R20	SP1 - Functional completeness, SP9 - Confidentiality, SP10 - Integrity, SP12 - Modularity, SP13 - Testability	AE, PA	C12
FR07	Intuitive interface with HITL integration	RAIDO must feature an interactive, intuitive interface with HITL support for user feedback. The system must provide XAI features, visualization of model performance, and decision-making insights.	R08, R24, R25, R28	SP6 - Operability, SP7 - User engagement, SP2 - Functional correctness, SP12 - Modularity, SP14 - Adaptability	AE, PA	C07, C09
FR08	Seamless integration with legacy systems	RAIDO should support plug-and-play integration with existing user infrastructure (e.g., dataset collection systems, AI models). The system should require minimal configuration to ensure smooth interoperability with legacy systems and applications.	R09, R33	SP1 - Functional completeness, SP5 - Interoperability, SP6 - Operability, SP12 - Modularity, SP14 - Adaptability	AE, DE, PA	C09

5 Conclusions

The "Design of the General RAIDO Platform and Architecture" deliverable marks a significant milestone in the RAIDO project, providing the blueprint for a comprehensive and innovative architecture designed to address the evolving challenges of Industry 4.0. The document outlines the development of RAIDO's architecture, emphasizing the orchestration of computational resources, AI models, and data to optimize system performance and resource efficiency. Central to this design is the commitment to energy-efficient, green-AI solutions that contribute to global sustainability efforts, making RAIDO a crucial component in the future of intelligent, scalable, and eco-conscious AI systems.

The document begins with a high-level overview of the RAIDO architecture in Section 2, followed by detailed explorations in subsequent sections (cf. Section 3) of the key components and their functions, such as AI pipelines, data management, and green-AI orchestration. Section 4's Requirements Traceability Matrix ensures that the design is closely aligned with both functional and non-functional requirements, demonstrating how stakeholder and user needs are directly integrated into the technical components of the architecture.

As we progress, this document sets the foundation for the realization of a fully integrated, sustainable, and efficient RAIDO platform. The architecture presented here will evolve and be refined as the project advances, leading to the deployment of RAIDO's innovative solutions within edge and cloud infrastructures, as outlined in future deliverables.

RAIDO is poised to drive the next generation of AI and data optimization systems, supporting green-AI technologies and transforming industrial operations through its comprehensive, interconnected, and sustainable platform.

6 References

[1] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. ICML 2021.

[2] Zhao, B., Cui, Q., Song, R., Qiu, Y., & Liang, J. (2022). Decoupled Knowledge Distillation. CVPR 2022.

[3] Wu, K., Peng, H., Zhou, Z., Xiao, B., Liu, M., Yuan, L., Xuan, H., Valenzuela, M., Chen, X. (S.), Wang, X., Chao, H., & Hu, H. TinyCLIP: CLIP Distillation via Affinity Mimicking and Weight Inheritance. ICCV 2023 .

[4] Bulat, A., & Tzimiropoulos, G. LASP: Text-to-Text Optimisation for Language-Aware Soft Prompting of Vision & Language Models. CVPR 2023.

[5] Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. ICLR 2023.

[6] M. T. Hagos et. al, (2022). "Impact of Feedback Type on Explanatory Interactive Learning", Intern Symposium on Methodologies for Intelligent Systems, https://doi.org/10.1007/978-3-031-16564-1_13 .

[7] D. Minh et al., (2021). "Explainable artificial intelligence: a comprehensive review", Artificial Intelligence Review, https://doi.org/10.1007/s10462-021-10088-y .

[8] C. Meske et al. (2022). "Explainable artificial intelligence: objectives, stakeholders, and future research opportunities", Information Systems Management 39.1, https://doi.org/10.1080/10580530.2020.1849465 .

[9] M. R. Islam et al. (2022). "A systematic review of explainable artificial intelligence in terms of different application domains and tasks", Applied Sciences 12.3, https://doi.org/10.3390/app12031353 .

[10] D. Tchuente et al. (2024). "A methodological and theoretical framework for implementing explainable artificial intelligence (XAI) in business applications", Computers in Industry 155:104044, https://doi.org/10.1016/j.compind.2023.104044 .

[11] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, 267, pp.1-38, https://doi.org/10.1016/j.artint.2018.07.007 .

[12] Lundberg, S. M., & Lee, S. I. (2017). "A unified approach to interpreting model predictions." In Advances in Neural Information Processing Systems (pp. 4765-4774). https://doi.org/10.48550/arXiv.1705.07874

[13] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA, (pp.1135-1144). https://doi.org/10.1145/2939672.2939778

[14] Doshivelez, A., & Kim, P. (2017). "Towards a rigorous framework for Interpretable Machine Learning." Proceedings of the 34th International Conference on Machine Learning (Vol. 70, pp. 678-687). arXiv:1702.08608

[15] Bach, S., et al. (2015). "On pixel-wise explanations for non-linear classifier decisions by means of Shapley values." PLOS ONE, 10(7), e0130140. https://doi.org/10.1371/journal.pone.0130140

[16] Montavon, G., et al. (2018). "Explaining deep neural networks with layer-wise relevance propagation." In Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (pp. 193-209). Springer, https://doi.org/10.1007/978-3-030-28954-6_10 .

[17] Wachter, S., Mittelstadt, B. and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL & Tech., 31, p.841 https://doi.org/10.48550/arXiv.1711.00399 .

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2020. Grad-CAM: Visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vis. 128, 2 (2020), 336–359. https://doi.org/10.1007/s11263-019-01228-7

[19] Zuluaga, M., et al. “Active Learning for Multi-Objective Optimization,” in Proceedings of the 30th International Conference on Machine Learning, PMLR 28(1):462-470, 2013, DOI: 10.1145/3042817.3042991.

[20] Christiano, P. F., et al. “Deep reinforcement learning from human preferences,” in Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4302–4310, 2017, DOI: 10.5555/3295222.3295344.

[21] Snoek, J., et al. “Practical Bayesian optimization of machine learning algorithms,” in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 2951–2959, 2012, DOI: 10.5555/2999325.2999464.

[22] Pfeiffer, J., et al. “Modular deep learning,” arXiv preprint arXiv:2302.11529v2, 2023.

[23] Testi, M., et al. “MLOps: A Taxonomy and a Methodology,” IEEE Access, 2020, DOI: 10.1109/ACCESS.2022.3181730.

[24] Tashan, W., et al. “Mobility Robustness Optimization in Future Mobile Heterogeneous Networks: A Survey,” IEEE Access, 10, 2022, DOI: 10.1109/ACCESS.2022.3168717.

Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101135800