Funded by the European Union

FALCON is funded under the Horizon Europe Framework Program
Grant Agreement ID 101121281

FALCON
Fighting Corruption & Organised Crime

Deliverable D3.2
Title: FALCON Framework Architecture

Dissemination Level:	PU - Public
Nature of the Deliverable:	R - Document, report
Date:	31/05/2024
Work Package:	WP3 - Anti-corruption AI framework co-design
Editors:	UPV
Reviewers:	VICOM, UCSC
Contributors:	ICCS, IOSB, ENG, CENTRIC, CERTH, SPH, VICOM, BPTI

Abstract: This deliverable outlines the architectural design and methodological approach for the FALCON system to enhance anti-corruption efforts. It details functional and non-functional specifications, employs an agile methodology, and integrates a CI/CD pipeline for continuous improvement and testing. The document covers high-level architecture, communication, authorization, deployment strategies, and emphasizes data security and trustworthy AI principles. Each tool's role and functionality within the FALCON toolkit are described, providing a cohesive development framework.

Disclaimer

Revision History

Date	Rev.	Description	Partner(s)
29/04/2024	0.1	Document template	UPV
08/05/2024	0.2	TOCs review and definition of section 7	UPV
10/05/2024	0.3	Sections 1,2 Content	UPV
16/05/2024	0.4	Added comments and format correction, Changes in 4.2.2 and added Section 5.2	IOSB
20/05/2024	0.5	Added contributions. Sections 5.2, 6 and 7.	IOSB, ICCS, VICOM, ENG, CERTH, SPH
21/05/2024	0.6	Added more tools in Section 7	UPV
22/05/2024	0.7	Added CENTRIC Contributions, Format corrections	CENTRIC, UPV
24/05/2024	0.8	Added BPTI Contributions	BPTI
27/05/2024	0.9	Format and language corrections	UPV
29/05/2024	0.91	Review comments integration	VICOM
30/05/2024	0.92	Review comments integration	UCSC
30/05/2024	0.93	Format corrections, images alignment, final release generation	UPV
31/5/2024	1.0	Review by the Coordinator - Submission	ICCS

List of Authors

Partner	Author
UPV	Francisco Pérez, Alberto García
IOSB	Christian Ellmauer, Dirk Pallmer
CERTH	Kostas Loumponias, Ourania Theodosiadou, Vassilis Solachidis, Nicholas Vretos, Fotini Dougali
SPH	Marios Zacharias, Kostas Tripolitis
VICOM	Xabier Etxeberria, Francesco Zola
ICCS	Evgenia Adamopoulou, Nikolaos Peppes, Theodoros Alexakis, Emmanouil Daskalakis
CENTRIC	Chris Guiver, Ethan Collins, Geraldine Meloy
ENG	Marco San Biagio
BPTI	Kostas Griška

Internal Reviewers

Partner	Reviewer
VICOM	Xabier Etxeberria, Francesco Zola
UCSC	Caterina Paternoster, Andrea Carenzo

Table of Contents

Index of Figures

Index of Tables

Glossary

2FA	Two factor Authentication
AI	Artificial Intelligence
AML	Adversarial Machine Learning
API	Application Programming Interface
BCP	Border Control Point
CRM	Common Representational Model
DMP	Data Management Plan
DoA	Description of Action
CI/CD	Continuous Integration and Continuous Delivery
CRM	Common Representational Model
DNNs	Deep Neural Networks
EC	European Commission
EEAB	External Expert Advisory Board
ENISA	European Union Agency for Cybersecurity
ETL	Extract, Transform and Load
FCT	Fighting Crime and Terrorism
FPA	Falcon Predictive Analytics
FTD	FALCON Trend Detection
HMI	Human Machine Interface
IP	Intellectual Property
IPRs	Intellectual Property Rights
ISO	International Organization for Standardisation
JWT	JSON Web Token
KB	Knowledge Base
LEA	Law Enforcement Agency
ML	Machine Learning
MoM	Minutes of Meeting
MVCS	Model View Controller Service
OWL	Web Ontology Language
RBAC	Role Based Access Control
RDF	Resource Description Framework
REST	Representational State Transfer

RKE2	Rancher Kubernetes Engine Government
RNNs	Recurrent Neural Networks
SAB	Security Advisory Board
SAI	Securing Artificial Intelligence
SDMA	Secure Data Management and Audit Trail
SQL	Structured Query Language
SPARQL	SPARQL Protocol and RDF Query Language
SPEL	Societal, Privacy, Legal, Ethics
TDB	Triplestore Database
UI	User Interface
VM	Virtual Machine
WP	Work Package
XML	Extensible Markup Language

Executive Summary

The FALCON project, funded by the Horizon Europe Framework Program, aims to develop a global framework to enhance anti-corruption efforts through advanced technology integration. This deliverable, D3.2 "FALCON Framework Architecture", outlines the architectural design and methodological approach to develop the FALCON system. It builds on the fundamental requirements established in D3.1 "Use Cases and Requirements" and lays the foundation for subsequent development phases. The primary objective of this deliverable is to present the detailed architectural framework of the FALCON system, covering functional and non-functional specifications validated by FALCON stakeholders to ensure that they meet the project objectives and user requirements. The deliverable also outlines the implementation process, focusing on the design of the core elements of the toolset and their interconnections, with emphasis on data exchange processes and human-machine interface (HMI) design.

The development of the FALCON framework employs an agile methodology, characterized by its iterative process, flexibility, and responsiveness to changing requirements. This approach facilitates continuous improvement and ensures that development remains aligned with user needs and project objectives. The integration of a continuous integration and delivery (CI/CD) process using GitLab CI/CD further enhances this process, enabling continuous updates and robust testing of all components. The deliverable provides a high-level overview of the FALCON framework, detailing its major components, external interfaces, and platform requirements, including user requirements, functional and non-functional specifications, and sequence diagrams to illustrate interactions and data flows within the system.

An important part of the document is dedicated to communication, authorization, and deployment strategies for the FALCON platform, including a tool interconnection matrix, data exchange models, authentication mechanisms and deployment strategies. The deployment strategy leverages modern infrastructure technologies such as Kubernetes to ensure scalability, security, and robustness. The deliverable also addresses secure data management within the FALCON platform, outlining data management requirements and describing the FALCON Knowledge Base architecture, which integrates several data sources and supports complex analytical tasks.

Incorporating trustworthy AI principles is a key focus of the FALCON project, and the deliverable discusses measures to ensure that AI components are designed and deployed in a secure, reliable, and ethical manner. This commitment to responsible AI use is critical for maintaining the integrity and effectiveness of the FALCON platform. Each tool within the FALCON toolkit is described in detail, including its roles, functionalities, and requirements, providing a granular look at the technological

components that make up the FALCON platform and illustrating how they work together to achieve the project's objectives.

1. Introduction

1.1 Purpose of the Deliverable

This document aims to present the architectural framework of the FALCON system, developed from the user requirements, and use case specifications received in Task 3.1. It details the functional and non-functional specifications validated by FALCON stakeholders, as well as the detailed architecture of the system and planned information exchanges. The deliverable serves as a foundational blueprint outlining how the core elements of the FALCON toolkit are designed and interconnected, emphasizing data exchange processes and HMI design. By articulating this framework, the deliverable facilitates a shared understanding among technical and non-technical partners involved in the project, ensuring that all subsequent development efforts are aligned with the established specifications and stakeholder expectations. Ultimately, this document will guide the implementation process within the project, setting the stage for the development of a robust and effective system.

1.2 Relevance of D3.2 and Connections with other Work Packages

The Deliverable D3.2, "FALCON Framework Architecture," is strategically positioned within the FALCON project, serving as a link between the foundational requirements outlined in D3.1 "Use Cases and Requirements" and the upcoming development phases encapsulated in Deliverables D3.3 "FALCON Prototype R1.0," D3.4 "FALCON Prototype R2.0," and D3.5 "Final FALCON Prototype." D3.1 provided a comprehensive list of functional, operational, security, and communication requirements necessary for the development of the FALCON infrastructure and tools. Building on this groundwork, D3.2 defines the system architecture and development methodologies that will shape the implementation of these requirements. It establishes a detailed design of the FALCON toolkit, including the interactions between its components and the data exchange protocols, which are essential for ensuring that the prototypes developed in subsequent phases are robust, secure, and efficient.

Moreover, D3.2 lays the architectural foundation that will directly influence the work packages WP4 "Corruption Data Acquisition and Analysis Tools" and WP5 "Risk Assessment, Investigation, and Decision Support Tools." By defining the framework architecture and the characteristics of the tools within the FALCON framework, D3.2 ensures that the technological solutions developed in these work packages are not only aligned with the initial specifications but are also capable of integrating seamlessly, supporting the project's overall goals of enhancing detection, analysis, and decision-making capabilities in anti-corruption efforts, maintaining trustworthy AI system regulations presented by the EU. This deliverable thus acts as a blueprint, guiding the development, integration, and deployment of the FALCON tools across different stages

and work packages, ensuring consistency and functionality throughout the project lifecycle.

1.3 Structure of the Deliverable

This deliverable is structured to present a detailed and coherent view of the FALCON framework architecture, ensuring a comprehensive understanding of its components and their interactions. The document begins with an Introduction in **Section 1**, which includes a discussion on the purpose of the deliverable, its relevance within the broader project context, and its connection to other work packages and deliverables. This section sets the stage for the deliverable by providing essential background information and outlining the scope of the framework.

Section 2 focuses on the Development Methodology, describing the approaches and techniques employed to develop the architecture of the FALCON framework. This section details the processes and standards adhered to during the framework's formulation, ensuring a clear understanding of the development methodologies employed.

In **Section 3**, a High-Level View of the FALCON Framework Architecture is provided, starting with a general overview, and followed by specifics on External Interfaces and Platform Requirements, including User, Functional, Non-Functional Requirements, and Sequence Diagrams. This comprehensive breakdown facilitates an in-depth understanding of how the framework functions and integrates with external systems.

Section 4 provides a deeper dive into FALCON communication, authorization, and deployment, detailing the tool interconnection matrix, data exchange models, authentication mechanisms, and deployment strategies. This section is crucial to understanding how the framework maintains secure communication and data integrity.

Section 5, FALCON Secure Data Management and Knowledge Base, outlines the data management requirements including user, functional, non-functional requirements, and sequence diagrams. It provides specifics on how data is managed securely within the framework, ensuring compliance and protection.

Section 6 explores the incorporation of Trustworthy AI principles within the FALCON project, discussing how AI components are designed to be reliable and ethical, reflecting the project's commitment to responsible AI use.

In **Section 7**, a Functional Description of the FALCON Tools is given, covering each tool provided by all tool providers with descriptions, user requirements, functional and non-functional requirements, and sequence diagrams. This section offers a granular look at the tools that comprise the FALCON toolkit, their roles, and their operational parameters.

The deliverable concludes with **Section 8**, Summary and Conclusions, summarizing the findings and outlining the conclusions drawn from the development and analysis activities. This final section highlights the key outcomes and provides a clear closure to the document, reinforcing the deliverable's objectives and achievements.

2. Development Methodology

For the successful development of the FALCON project and aligned with the activities to be performed during the T3.5 execution, an agile methodology will be utilized. This approach is characterized by its flexibility, iterative process, and ability to adapt to changing requirements over the course of the project. The Agile methodology is particularly well-suited for projects like FALCON, which involve complex systems and the integration of various technological components and user requirements.

One of the key advantages of agile methodology is its focus on continuous improvement and responsiveness to feedback. By organizing the development process into sprints—short, consistent cycles of development—project teams can evaluate the results of each iteration and adjust their strategies and plans based on real-time feedback from stakeholders and end-users. This iterative process not only enhances the quality and relevance of the developed system but also ensures that the project remains aligned with user needs and the project's overarching goals.

Furthermore, agile promotes regular communication and collaboration both within the development team and with external stakeholders. This ongoing interaction helps to clarify requirements and resolve potential issues early in the development process, reducing the risk of significant overhauls or changes at later stages.

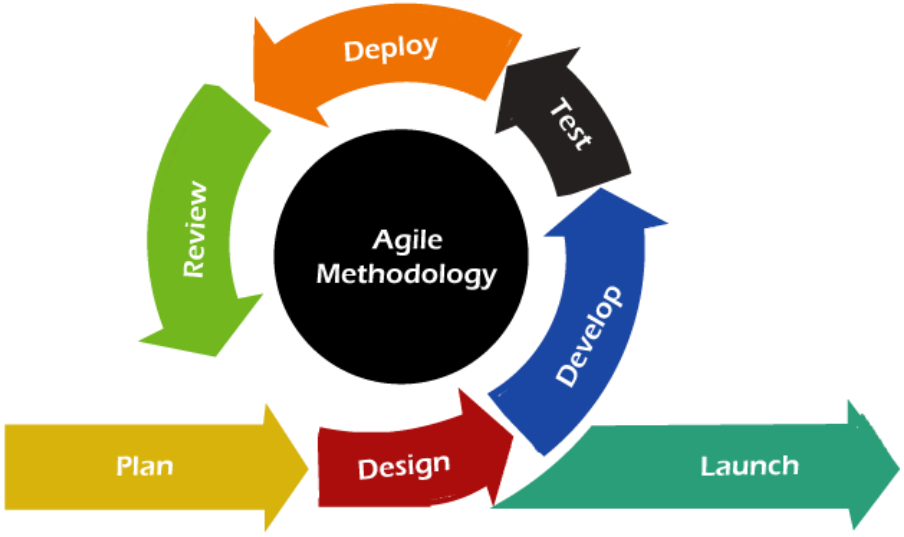

Figure 1. Agile Methodology Workflow

Implementing agile methodology in the FALCON project will facilitate a more efficient and effective development process, fostering innovation and ensuring that the final deliverables are both high-quality and closely aligned with the project's objectives. This approach is essential for managing the complexities of designing a framework like FALCON, which requires seamless integration of diverse technologies and consistent adaptation to emerging challenges in the field.

Once the methodology to be used for the FALCON project is in place, the integration plan will be developed. This plan is crucial, as it describes how the various components and services of the FALCON framework will be systematically combined and tested to ensure that they work as a cohesive unit. The integration strategy is designed to support the agile development process by enabling incremental integration and testing of components. This approach not only ensures that each part of the system works independently, but also verifies that they work together seamlessly within the broader ecosystem. The plan will encompass the methodologies, tools, and sequences for integrating the modular components developed by different teams, ensuring alignment with project goals and timelines. This step is vital to move from individual development activities to a fully functional system ready for deployment and implementation in the proposed use cases. Table 1 details the different phases and actors in the integration plan.

Table 1. Integration Plan

Iteration	Integration Point	Components[1]	Partners	Date
1st Dev Phase	Architecture and data connector design, first development phase of software component, including unit tests and integration tests. Installation and configuration CICD Platform. Deliver first Falcon platform prototype.		All technical Partners involved in WP3, WP4, WP5 and WP6	M5-M16
1st Dev Iteration	Beta release of some components			M5-M12
MS3	First tools available			M12
1st Platform Integration	Functional Tests, End-to-End integration tests			M12-M16
Prototype R1.0	First FALCON platform release			M16
2nd Dev Phase	Main development phase. Deliver second Falcon platform prototype.		All technical Partners involved in WP3, WP4, WP5 and WP6	M16-M26
2nd Dev Iteration	All components must be released in a stable form along with their corresponding unit tests			M16-M18
MS4	First series of pilots executed			M18

[1] Components column will be updated in following releases of the deliverable. All the identified components of each integration phase will be compiled in this table.

2nd platform integration	Functional tests – End-to-end integration tests ready for second release.			M18-M24
MS5	Improved tools and CIP available			M24
Prototype R2.0	Second FALCON platform release			M26
3rd Dev Iteration	All components should be released in a stable version, accompanied by their corresponding unit tests, and incorporate any feedback received from the initial pilots		All technical Partners involved in WP3, WP4, WP5 and WP6	M26-M28
MS6	Second series of pilots executed			M28
3rd Platform Integration	Last End-to-end integrations tests ready for the final release.			M28-M33
Final Prototype (MS7)	Final FALCON platform release			M33

As outlined in Table 1, the **first phase** spans from Month 5 to Month 16 and is focused on the design of the architecture and its components based on the user requirements defined in D3.1 "Use Cases and Requirements." This phase primarily involves developing connectors and adapters for accessing datasets and integrating various heterogeneous data sources, developing the middleware, and establishing a common representational model. The setup and configuration of the CI/CD platform will also take place during this phase, as well as the rollout of the first beta versions of the analysis tools, along with unit and integration tests to validate their functionality on the platform. This phase will culminate with the first release of the FALCON platform, documented in D3.3 "Prototype R1.0," at Month 16.

The **second phase**, spanning from Month 16 to Month 26, marks the primary development stage of the project. During this period, all FALCON tools will be advanced to a stable release. The system will harness innovative techniques to analyse corruption indicators from various data types and sources. Key activities include extracting indicators across different formats, implementing anomaly detection strategies, developing deep learning models to generate corruption alerts, and providing tools for spatio-temporal analysis.

Additionally, the platform will be enriched with several specialized applications: a corruption risk assessment tool, an investigative application tailored to handle corruption cases, predictive analytics capabilities, and a user-friendly dashboard for interpreting data. A continuous improvement module will also be integrated to enhance corruption intelligence continuously.

This phase is also crucial for operational testing, as the first pilot tests of the platform will be conducted. These tests are essential for validating the platform's functionality, collecting user feedback, identifying bugs, and making necessary enhancements to optimize the system's performance. Finally, this phase will conclude with the second release of the FALCON platform, marking a significant milestone in the project's timeline.

The **third and final phase** will extend from Month 26 to Month 33, represents the culmination of the FALCON development cycle. This third phase is dedicated to refining and perfecting all tools developed during the project. Every feature and functionality will be meticulously reviewed to ensure that any bugs are resolved, and all user feedback collected from the earlier pilot tests is effectively integrated.

This period is critical for achieving seamless integration across the entire FALCON platform. The tools must not only function optimally on an individual basis but also demonstrate flawless interaction and data communication with other system components. This holistic integration is essential for ensuring the platform's robustness and operational efficiency. In addition to technical refinement, this phase focuses on final performance optimizations and usability enhancements. The aim is to deliver a user-friendly, fully integrated system that meets the high standards set by the project goals and stakeholder expectations.

The phase will conclude with the final release of the FALCON platform. This release will embody the full capabilities of the system, showcasing the comprehensive efforts of the development team and stakeholders in creating a sophisticated tool for combating corruption. This final rollout is not just a technical milestone but also a strategic one, setting the stage for the platform's deployment and operational use in real-world scenarios.

To enhance the agile framework adopted by the FALCON project, a robust CI/CD pipeline will be implemented using GitLab as the primary platform. GitLab will support seamless updates and efficient integration of code changes across the project, enabling a streamlined workflow for continuous improvement of the project components. The introduction of a well-defined testing framework is another critical component of our methodology. This framework will include both automated and manual testing strategies to ensure comprehensive quality assurance at every stage of development.

For automated testing, established tools will be deployed such as Jest, Playwright, or Apache JMeter, which are known for their robustness and flexibility in handling various testing scenarios. These tools will allow to automate repetitive tasks and perform extensive regression testing to quickly identify any disruptions caused by new changes. Manual testing will complement these automated processes, focusing on complex scenarios that require nuanced human judgment and interaction, thereby providing a holistic approach to our testing efforts. This dual approach to testing will ensure that each release is thoroughly vetted for both functionality and performance, adhering to the high-quality standards set for the FALCON project.

To further enhance the robustness of our CI/CD pipeline within the FALCON project, each of the services or modules will be containerized using Docker. This containerization strategy allows for modular development and testing, facilitating isolated testing environments for each component without the risk of cross-contamination or dependency conflicts. By encapsulating each module in its Docker container, we ensure a uniform, scalable, and easily manageable deployment process.

Following the successful completion of both unit and integration tests, these Dockerized modules will be automatically deployed using Kubernetes. Kubernetes allows us to orchestrate and manage multi-container applications at scale, streamlining the deployment process across the different services of the FALCON platform. This system enhances our ability to handle automatic scaling, load balancing, and management of containerized applications, providing robust, production-ready deployment capabilities that are essential for the seamless operation of the FALCON infrastructure.

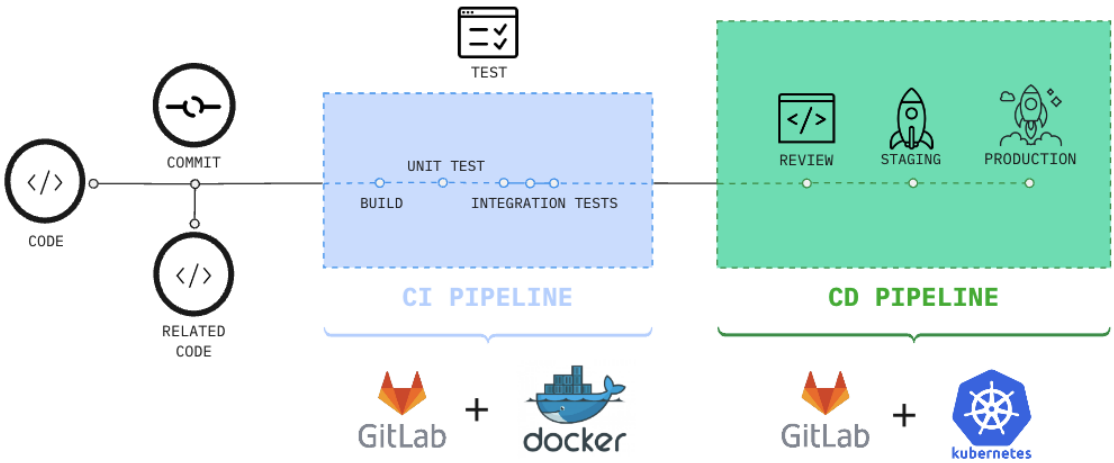

Figure 2. CI/CD Platform

As part of the agile process, the integration and testing phases will be closely aligned with the developments in Work Packages WP4 and WP5, where various tools and components will be developed. These efforts will be continuously refined and detailed in

Task T3.5 "Continuous Testing and Integration." This task will delve deeper into the specific methodologies, tools, and protocols we are using, ensuring that our approach to quality and performance aligns with the project’s long-term goals and delivers a reliable, effective product.

3. High level view of FALCON Framework Architecture

3.1 General View

The FALCON Framework Architecture is strategically developed to enhance the capabilities of anti-corruption and organized crime agencies through advanced technological integration. This architecture serves as the backbone for an integrated system that supports complex data analyses, ensuring both flexibility and robustness in its applications. It has been designed to handle vast amounts of data, facilitating real-time processing and comprehensive analytics that are crucial for quick decision-making in law enforcement operations. The design also prioritizes modularity and scalability, allowing for future enhancements and integration with emerging technologies. The complete conceptual view of the FALCON Platform architecture can be viewed in Figure 3. Most data exchanges between the different FALCON tools will be orchestrated by the FALCON Message broker described in Section 4.2 allowing the exchange of JSON messages following the FALCON data-model described also in same section. Additionally, Table 2 provides a concise summary of the FALCON Architecture Block Diagram, including a brief description that delves into the specifications, functionalities, and the comprehensive suite of tools that comprise the platform's pipeline, which will be detailed further in Section 7.

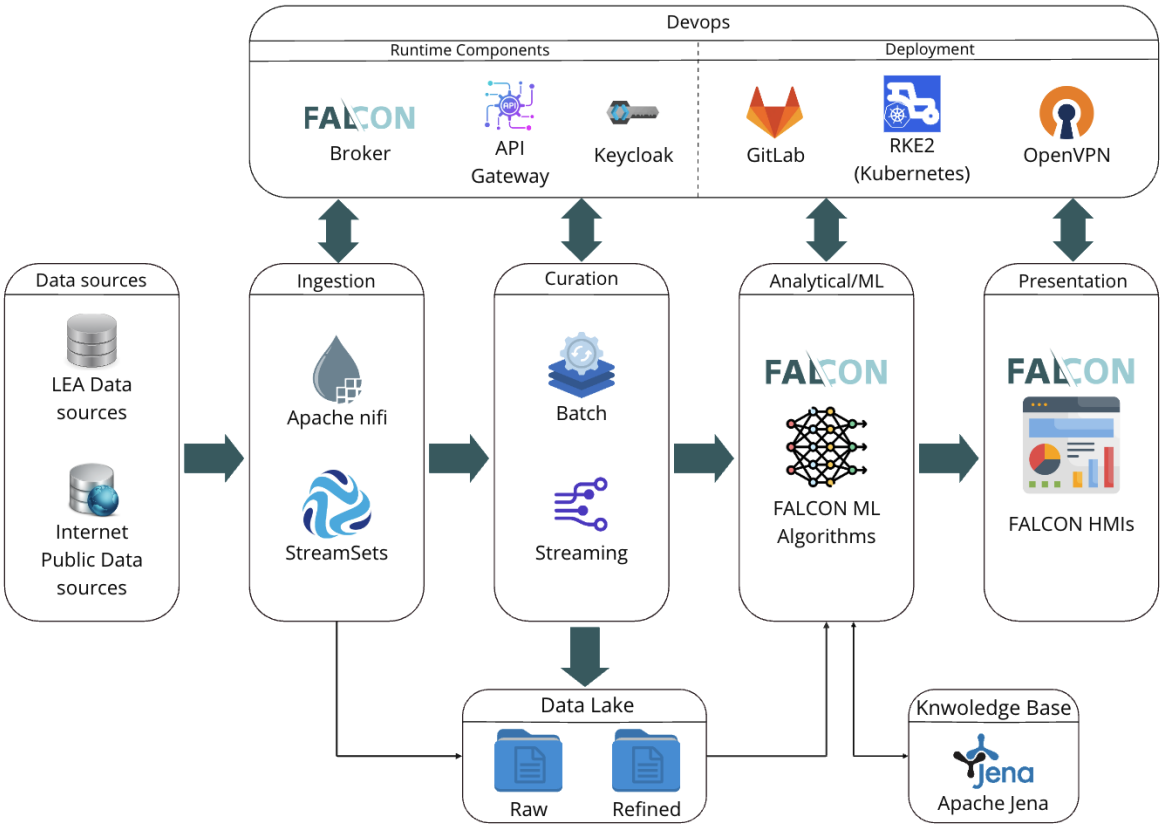

Figure 3. FALCON High Level Architecture

Table 2. Overview of FALCON Platform Blocks

Components Block	Related WP	Brief Description
DEVOPS	WP3	Central to ensuring the continuous integration and deployment of the FALCON platform, the DEVOPS block automates and streamlines the software development processes. It supports rapid iteration and robust quality assurance practices by managing the lifecycle of application development from coding through testing, building, and deployment.
Data Sources	WP4	This block serves as the repository of all data inputs into the platform. It includes external and internal data sources ranging from public databases and open-source data to sensitive law enforcement data, ensuring a comprehensive dataset is available for analysis.
Ingestion	WP3/WP4	Responsible for the initial collection and import of data from the diverse data sources identified. The ingestion block ensures that data is accurately and efficiently loaded into the system, supporting real-time and batch processing capabilities.
Curation	WP4	Focuses on the quality and usability of data. This block cleans, enriches, and verifies data to ensure its integrity and relevance. It also handles the harmonization and standardization processes, making data ready for analysis and storage in the Data Lake.
Analytical/ML	WP4	This block incorporates advanced analytics and machine learning models to extract insights and patterns from curated data. It supports a variety of analyses, from predictive modelling and anomaly detection to deep learning applications, all aimed at enhancing decision-making processes.
Presentation	WP5	Focuses on the visualization and reporting tools that make insights accessible and actionable to end-users. This block designs and implements dashboards, reports, and real-time alerts that facilitate easy interpretation and prompt action based on the analytics results.
Data Lake	WP3/WP4	Acts as a centralized repository that allows for the storage of structured, semi-structured, and unstructured data at scale. The Data Lake supports big data and analytics capabilities, providing a flexible environment for data exploration, analysis, and sharing across the platform.

3.2 External Interfaces

This section outlines the integration strategies for a diverse array of external data sources into the FALCON platform, identified through a thorough analysis of use cases as detailed in D3.1 "Use Cases and Requirements." These data sources are crucial in supporting the platform's broad investigative and analytical capabilities. The selection

and integration of these datasets are fundamental activities that ensure the platform can effectively break data silos, as emphasized in the project's objectives.

Given the variety of data types and sources—from financial transactions and corporate databases to open-source intelligence and satellite imagery—specialized connectors and adapters are essential. These tools are designed to ensure seamless data integration, efficient processing, and easy accessibility within the FALCON platform. This section will also provide a summary of the datasets, detailing the technical specifications required for their integration, which are crucial for maintaining the integrity and functionality of the platform as it handles complex data interactions.

Table 3. FALCON Identified Data Sources

Data Source	Type of Connector/Adapter	Data Access Protocol
Opentender.eu - Procurement data	REST API connector	HTTPS, OAuth2
Company data - Orbis	REST API connector	HTTPS
Cryptocurrency transactions (Bitcoin, Litecoin, Monero)	Blockchain connector	P2P Network, Blockchain API
Interest declarations	Custom API connector	HTTPS, REST API
Sanctions Lists	REST API connector	HTTPS, REST API
Vessel data	Maritime data adapter	AIS, Satellite API
PEP data	Financial data API connector	HTTPS, REST API
Real Estate Data	API connector	HTTPS, REST API
Car details (model, make, year, license plate)	Vehicle data connector	REST API, SOAP
Copernicus Sentinel	Satellite data adapter	OPeNDAP, WMS
IMF - Corruption Perception Index	Economic data API connector	HTTPS, REST API
International Consortium of Investigative Journalism	Custom scraper and API connector	HTTPS, REST API
AIS Log Klaipéda	Maritime data adapter	AIS, Satellite API
OSINT / Social media and Websites - news	Web scraper and social media connector	HTTPS, REST API, OAuth2
Worldbank: Global Public Procurement Database	API connector	HTTPS, REST API
International border transit records	Custom API connector	HTTPS, REST API
Border Guard Deployment and Schedule VSATIS	Custom API connector	HTTPS, REST API
Car model recognition	Image recognition service	HTTPS, REST API

Criminal History and Identification	Law enforcement data connector	HTTPS, REST API
Vehicle Registration and ownership	Vehicle registry API connector	REST API, SOAP
Road cameras data	Video data adapter	RTSP, HTTPS

After establishing how external data sources will be interfaced with the FALCON platform, it is crucial to discuss the underlying **integration considerations** that ensure the system is robust, secure, and scalable. These considerations are pivotal in designing an architecture that not only supports the current operational needs but is also flexible enough to adapt to future challenges and expansions.

1. **Security and Compliance**

Ensuring the security of data transactions and storage is paramount, especially given the sensitive nature of the data involved. Each connector and adapter must employ state-of-the-art security protocols such as TLS/SSL for data transmission and AES for data at rest. Compliance with international data protection regulations, like GDPR and HIPAA, is integral and must be built into the system from the ground up. This involves implementing robust authentication and authorization mechanisms, regularly updating data protection measures to address new vulnerabilities, and maintaining comprehensive audit logs for security monitoring and regulatory compliance.

2. **Data Handling Efficiency**

Efficient data handling is critical to manage large volumes of data from various sources without performance degradation. This includes implementing data caching strategies to improve response times, utilizing data compression techniques to minimize bandwidth usage, and adopting parallel processing and distributed computing techniques to enhance data processing capabilities.

3. **Scalability**

The architecture must support not only current data integration needs but also be adaptable to future expansions. This scalability can be achieved through a microservices architecture, containerization technologies like Docker and Kubernetes, and cloud-based data storage and processing solutions that can dynamically adjust resources based on system load.

4. **Data Normalization**

Data normalization is essential when dealing with data from various sources that often come in different formats. Developing a common data model for all incoming data

before it is processed and analysed ensures that data from different sources can be integrated seamlessly. Employing ETL (Extract, Transform, Load) processes and utilizing AI and ML techniques to automate the detection and correction of data discrepancies are key steps in this process.

5. **Error Handling and Recovery**

Robust error handling and recovery mechanisms are crucial to maintaining system reliability and availability. This includes implementing failover mechanisms to ensure that backup systems can quickly take over in case of a system failure, developing comprehensive error logging and notification systems to alert administrators to issues, and designing retry logic and data validation checks to handle transient errors and ensure data integrity during transmission and processing.

These integration considerations form the backbone of a reliable, secure, and efficient system, facilitating the FALCON platform's goal of becoming a robust tool in the fight against corruption.

3.3 Platform Requirements

In this section, a comprehensive overview of the platform-wide requirements for the FALCON project, categorizing them into functional and non-functional aspects based on the needs expressed by end-users is presented. This discussion sets the groundwork for a detailed analysis of how these requirements are implemented across the platform, offering insights into their status and priority levels. Unlike previous sections that will delve into tool-specific requirements, in Section 7, here we focus on the holistic attributes that the entire platform aims to achieve.

In this deliverable the user requirements are named and listed. You can examine each of them in more depth in D3.1 titled "Use Cases and Requirements".

Additionally, this section includes illustrative sequence diagrams that depict the operational scenarios outlined in D3.1. These diagrams serve as a visual guide to understanding the interactions and data flows within the platform, clarifying how different components integrate and cooperate to fulfil the defined use cases.

In the forthcoming analysis, while we will revisit and detail the specific requirements for each component of the platform, it's important to note that some requirements may be repeated across discussions. This repetition is deliberate, emphasizing how individual tools and features contribute to the realization of the overall platform requirements. By approaching the analysis from a comprehensive, platform-wide perspective, we aim to provide a clear and cohesive understanding of how the FALCON project is structured to meet the broad and varied demands of its stakeholders.

3.3.1 Functional Requirements

Table 4. Platform – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-05	System must be able to analyse and cross-reference different media formats efficiently.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-09	Support concurrent access by multiple users without degradation in system performance.	Must-have
FUN-10	System must provide multi-language support to accommodate users in different regions.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have
FUN-13	System should include ML capabilities that can be trained and improved over time.	Should-have
FUN-16	System must be compatible with the SIENA[2] System for extended functionalities.	Could-have

3.3.2 Non-Functional Requirements

Table 5. Platform – Non-functional Requirements

ID	Acceptance criteria	Ranking
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have
SEC-03	Enhanced security measures must be in place to protect sensitive data from unauthorized access.	Must-have

[2] SIENA stands for « Secure Information Exchange Network Application » and refers to a platform developed by Europol enabling the swift and user-friendly exchange of operational and strategic crime-related information among Europol member states, its liaison officers, analysts and experts as well as third parties with which Europol has cooperation agreements or working agreements. A more detailed description of SIENA can be found here: https://www.europol.europa.eu/operations-services-and-innovation/services-support/information-exchange/secure-information-exchange-network-application-siena

SEC-04	All data in transit and at rest must be encrypted; infrastructure must ensure data integrity and security.	Must-have
SEC-07	System must automatically save data at regular intervals or under specific conditions to prevent data loss.	Must-have
SEC-10	System must be robust and maintain operational capacity under various stress conditions.	Must-have
SEC-11	System must have comprehensive backup solutions to safeguard data integrity.	Must-have
SEC-13	Enhance security for user authentication to prevent unauthorized access.	Should-have
OPE-01	System must have failover capabilities and rapid recovery methods to ensure data integrity and availability.	Must-have
OPE-02	Interface must allow customization to meet the specific needs of different user types.	Must-have
OPE-04	Search functionality must be robust and capable of handling complex queries efficiently.	Must-have
OPE-08	Dashboard must update in real-time with full traceability and logging of all user actions and system responses.	Should-have
OPE-09	System must be designed to scale seamlessly as the amount of data and number of users grows.	Should-have
COM-01	System must support specified communication technologies to ensure wide accessibility and connectivity.	Must-have
COM-02	The system must facilitate secure and efficient communication between all relevant stakeholders.	Should-have
COM-03	System must ensure seamless integration with different IT environments and support various data formats.	Should-have
COM-04	Notifications must be timely and relevant, ensuring that all users are promptly informed of critical changes.	Should-have
COM-05	Collaboration tools must be secure, intuitive, and capable of supporting a multi-user environment.	Should-have
COM-06	The platform must be able to connect with various external systems and databases efficiently.	Should-have
COM-07	System should have mechanisms to alert users about system states, errors, or important notifications.	Should-have
COM-08	The system should support virtual communication tools embedded within the platform to facilitate discussions.	Could-have
COM-09	The system must provide interfaces and documentation in multiple languages to support international users.	Could-have

3.3.3 Sequence Diagrams

In the early stages of the FALCON Framework Architecture, it is essential to map out the interactions and data flows between the system's components clearly. Sequence diagrams serve this purpose by illustrating how processes interact within the platform, particularly focusing on data ingestion and the exchange of information between components. These visual representations are crucial for understanding the operational dynamics and for ensuring that all system components integrate seamlessly.

At this preliminary phase, several foundational sequence diagrams have been defined:

3.3.3.1 *Data Ingestion Process*

This diagram details the steps involved in ingesting data from various sources into the platform. It outlines the sequence of actions from data collection to data storage, highlighting the interaction between external data sources, the platform's ingestion services and the notifications through the platform Broker.

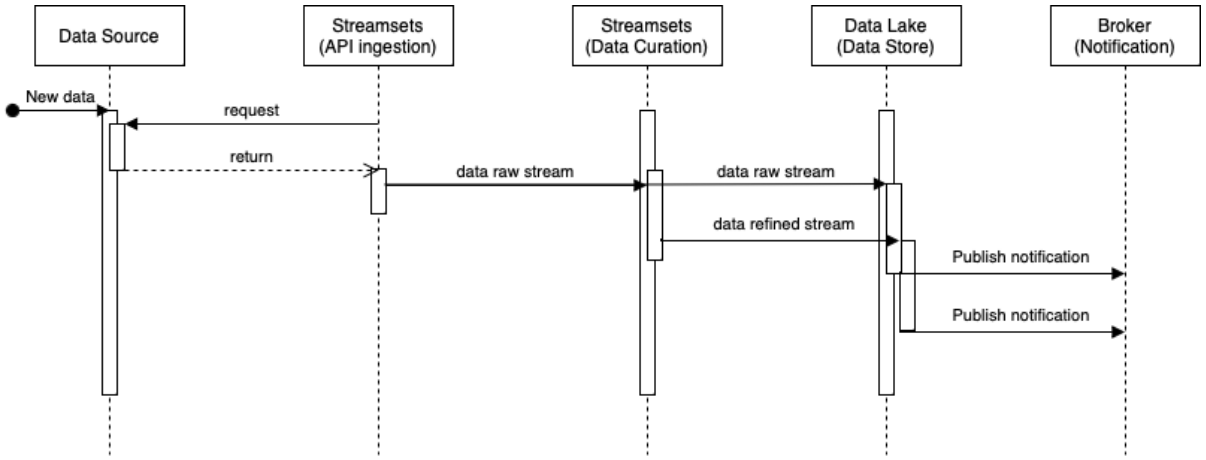

Figure 4. Data Ingestion Process Sequence Diagram

3.3.3.2 *Data Information Exchange (Through FALCON Broker)*

This sequence illustrates how data flows through the FALCON Broker, a central piece in managing internal communications. The diagram shows the publishing and subscribing mechanisms that facilitate data transfer between different microservices within the architecture.

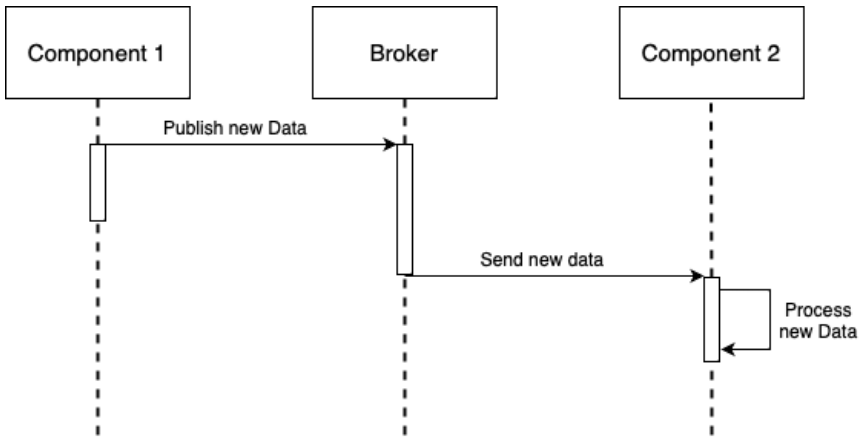

Figure 5. Data Information Exchange (Through FALCON Broker) Sequence Diagram

3.3.3.3 *Data Information Exchange (Through Component API)*

This diagram focuses on the interactions that occur via APIs between various components. It details the request and response cycles, showing how components communicate to perform specific functionalities and data processing tasks.

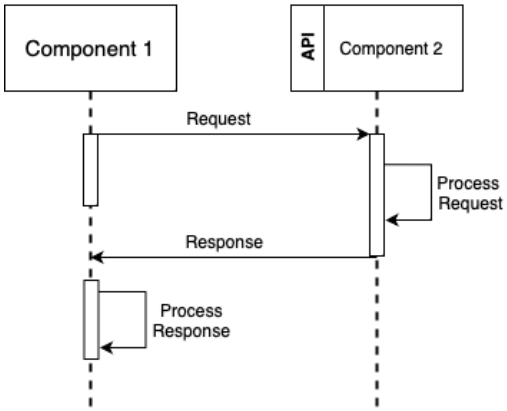

Figure 6. Data Information Exchange (Through Component API) Sequence Diagram

Given the FALCON platform's early development stage, these diagrams are subject to updates and revisions. The documentation of the FALCON prototypes (D3.3, D3.4 and D3.6) which will follow this document will include more detailed sequence diagrams reflecting any new components added to the system, changes in data flow, or modifications in component interactions. These updates will provide additional clarity and enhance understanding of the platform's evolving architecture.

- Base Diagrams: Currently, the diagrams serve as a baseline to establish the fundamental processes and interactions within the platform.
- Future Enhancements: As the platform develops and more detailed requirements and specifications are established, these diagrams will be enhanced to reflect more complex interactions and include additional components or processes as necessary.

The sequence diagrams are not only tools for visual communication within the development team but also serve as vital documentation for stakeholders to understand how the FALCON platform manages and processes data. By continually updating these diagrams, the FALCON project ensures that all participants and developers maintain a clear view of the system's operational procedures and are well-informed of any architectural changes.

4. FALCON Communication, Authorization and Deployment

4.1 Tools Interconnection Matrix

In the FALCON project, ensuring seamless communication and interaction between various tools and components is crucial. To manage and visualize these interactions effectively, a Tools Interconnection Matrix has been developed as it is shown in Figure 7.

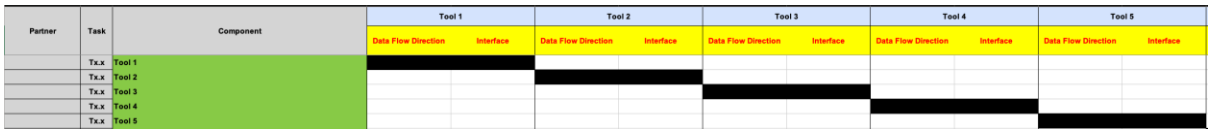

Figure 7. FALCON Tools Interconnection Matrix

This matrix serves as a comprehensive blueprint, detailing the relationships and data flows among the different components within the system. Here's an overview of how the matrix is structured and the vital information it contains:

4.1.1 Matrix Structure

- **Row Headers:** Each row header lists a specific tool or component involved in the FALCON project, along with essential details such as the partner responsible for developing or integrating that component, and the specific task or work package where the development or integration activity is scheduled.
- **Column Headers:** Similarly, the column headers replicate the list of tools and components. This setup allows the matrix to cross-reference every component with every other component to determine how they interact.

4.1.2 Contents of the Matrix

- **Development/Integration Partner**: This field specifies which project partner is responsible for the development or integration of the particular tool or component. This helps in coordinating efforts and clarifying responsibilities.
- **Task or Work Package:** This indicates the task or work package under which the development or integration of the tool/component is happening, aligning the development efforts with the project's timeline and deliverables.
- **Data Flow Direction:** For each pair of tools/components, the matrix specifies the direction of data flow:
    - Send: Where one tool sends data to another.
    - Receive: Where a tool receives data from another.
    - Exchange: Where there is a bidirectional exchange of data between tools.
- **Communication Interface**: This specifies the interface or protocol through which the communication occurs, such as REST APIs for web services, a message broker for event-driven architectures, or other custom interfaces designed for specific needs.

4.1.3 Purpose and Benefits

- **Visualization of Interactions:** The matrix provides a visual representation of how each component interacts with others, helping to identify potential bottlenecks or dependencies that could impact the system's architecture or performance.
- **Clarification of Responsibilities:** By associating each component with a specific partner and task, the matrix ensures clarity in terms of responsibility and accountability, which is crucial for effective project management.
- **Facilitation of Integration Testing:** Understanding the data flow directions and communication interfaces allows for more effective planning and execution of integration testing, ensuring that all components interact seamlessly as intended.
- **Support for Scalability and Modifications:** The matrix can be easily updated as new components are added or existing ones are modified, supporting the scalability of the project and making it adaptable to evolving requirements.

The Tools Interconnection Matrix is a dynamic document, continually updated as the project progresses and as components evolve. It serves not only as a planning tool but also as a critical reference for developers, project managers, and stakeholders throughout the lifecycle of the FALCON project. This proactive approach to documenting and managing interconnections ensures that the project remains organized, transparent, and on track to achieve its objectives.

4.2 Data Exchange & Data Model

4.2.1 Data Exchange

The FALCON platform employs sophisticated data exchange methods to manage interactions with multiple diverse data sources and internal services effectively. This intricate process is facilitated through two primary mechanisms: API calls and an internal communication broker, both of which are significantly enhanced by the integration of an API Gateway.

1. **API Calls:**

   External Communication: The platform leverages RESTful APIs for their scalability and flexibility, enabling FALCON to engage securely and efficiently with external data sources and services. These APIs are crucial for ensuring real-time access to data updates, which is vital for the platform's operational responsiveness and effectiveness.

   Standardization and Security: To ensure consistency and reliability in communication, all API interactions adhere to standardized protocols. This standardization supports uniform data handling and integration across different systems. Security is a paramount concern; hence, rigorous measures such as

OAuth for robust authentication and HTTPS for secure data transmission are implemented to protect data exchanges from unauthorized access and cyber threats.

2. **API Gateway:**

   Unified Access Point: The API Gateway is proposed to streamline interactions by acting as a single access point for all API calls to the microservices within the platform. This unification simplifies the management of these interfaces and enhances user experience by providing a coherent structure for data interactions.
   Security Enhancement: Beyond simplifying access, the API Gateway significantly boosts security. It incorporates advanced security features such as rate limiting, to prevent abuse and overuse of resources; IP whitelisting, to ensure that only authorized users can access certain services; and identity verification, to authenticate user identities rigorously. These features are essential in safeguarding the platform against common security threats and vulnerabilities.
   Simplified Interface for HMIs: The consolidation of API calls through the gateway provides the HMIs of the FALCON platform with streamlined access to various microservices. This centralization is critical for reducing complexity and enhancing the manageability and monitoring of API interactions, thereby improving the overall system accessibility and usability for end-users.

3. **FALCON Communication Broker:**

   Internal Communication Management: At the heart of internal communications, the broker plays a pivotal role in facilitating robust interactions between different components of the platform. It manages message routing, coordinates requests and responses, and orchestrates complex service interactions efficiently. This system ensures that all components can communicate their needs and responses within the platform without direct dependencies on each other.
   Decoupling and Reliability: Using a communication broker allows the platform to achieve greater decoupling between its components, which is crucial for maintaining system integrity during individual component upgrades or failures. This architectural choice enhances the platform’s fault tolerance and scalability. By enabling components to operate independently while still communicating effectively, the broker ensures that the system can scale without significant re-engineering.

Together, these advanced data exchange mechanisms ensure that the FALCON platform can handle the complexities associated with processing and analysing vast amounts of diverse data efficiently and securely. The use of a unified API gateway and a robust internal communication broker underpins the platform's ability to adapt to changing data demands and maintain high performance and reliability.

4.2.2 Data Model

In addition to the comprehensive data exchange mechanisms outlined in the FALCON platform, a sophisticated data model is integral for supporting effective communication and data management within the system. This model includes the use of a semantic model, the so-called Common Representation Model (CRM) that FALCON's ontological Knowledge Base is based on. The Knowledge Base collects and shares the evidence that has been created by the WP4 and 5 tools. For details about the Common Representation Model refer to Section 5.

Furthermore, to manage internal data workflows and extensive inter-service communications, the platform necessitates a robust system for documenting these interactions. Additional tabs have been added to the Integration Matrix spreadsheet (as shown in Figure 8), dedicated to service interactions, which plays a critical role in maintaining a centralized directory of all services and their interactions across the platform.

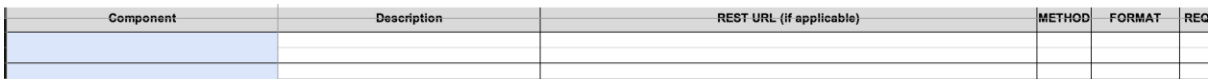

Figure 8. REST Services Information.

Here's how the information is structured within REST services tab:

- **Component**: Identifies the specific microservice or component within the FALCON platform.
- **Description:** Provides a brief description of the service or the data interaction facilitated by the component.
- **REST URL (if applicable):** Specifies the endpoint URL for services that interact through RESTful APIs.
- **METHOD:** Indicates the HTTP method used (GET, POST, PUT, DELETE) for API calls.
- **FORMAT:** Describes the data format used (e.g., JSON, XML) in the API interaction.
- **REQUEST PARAMETERS** (if applicable): Lists any parameters required to make the API call.
- **REQ/RES EXAMPLE**: Offers example request and response bodies to illustrate how data should be structured for API interactions.

For components communicating via the internal communication broker:

- **Component**: Specifies the component involved in the message exchange.
- **Description**: Provides details about the type of messages or data exchanged.
- **Topic**: Identifies the broker topic under which messages are published or subscribed.
- **Message Example:** Presents a sample message to demonstrate the structure and content expected in the communication.

This structured approach to documenting API services and broker messages ensures that developers and system integrators have clear and accessible references for how data should be exchanged within the FALCON platform. It also supports consistency and standardization in service implementation and integration, crucial for maintaining the system's reliability and efficiency.

4.3 Platform Authentication Mechanism

4.3.1 Keycloak SDMA Setup

Keycloak allows user and software modules alike to independently authorise using credentials and carry tokens of trust that verify their origin as coming from the Keycloak instance. Through the project, the SDMA moves parts of its authorisation structure externally to use the third-party authorisation of Keycloak. Rather than managing user credentials and sessions internally from within the SDMA, we verify JWT tokens, containing user details (e.g., ID, Role) on receipt of requests against the signed public certificate provided by Keycloak.

Keycloak will be employed to manage the authentication and authorisation in the FALCON platform. Such capabilities, however, rely mainly on JSON web tokens (JWT) to facilitate the secured data storage, access, and management, as well as audit trail. Briefly, JWT is a JSON object that provides secured representation of information between two parties, and more importantly, owns the authentication token and method in the Secure Data Management side of SDMA's API. Figure 9 provides an overview of a general workflow process of the Secure Data Management component of SDMA and its interactions with Keycloak.

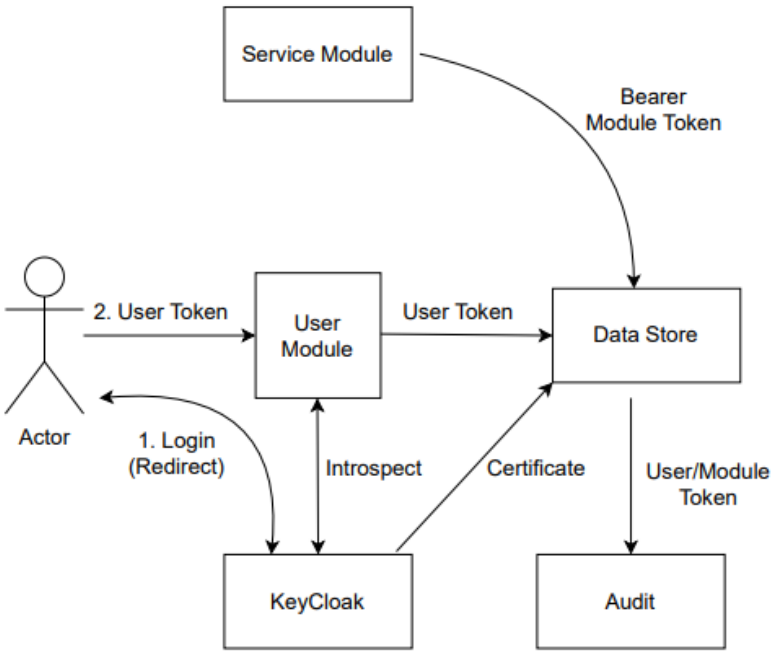

Figure 9. SDMA Workflow with Keycloak

4.4 FALCON Deployment

The deployment strategy for the FALCON platform is designed to be robust and flexible, accommodating the complex requirements of a highly integrated and scalable anti-corruption framework. This strategy leverages modern infrastructure technologies and follows best practices in CI/CD to ensure seamless, efficient, and error-free deployments.

Virtual Machines and Kubernetes Infrastructure:

- Virtualized Environment: The deployment architecture is based on a series of Virtual Machines (VMs), which provide a controlled and consistent environment for deployment. This setup ensures that the platform can be deployed in a standardized manner, reducing discrepancies between different deployment environments.
- Kubernetes Deployment: At the core of the deployment strategy is RKE2, a stable distribution of Kubernetes, which facilitates the orchestration and management of containerized applications. Kubernetes enables the deployment, scaling, and management of microservices across the infrastructure, ensuring that all components of the FALCON platform function cohesively and reliably.

CI/CD Philosophy:

- Continuous Integration and Delivery: Following the principles of CI/CD, the FALCON platform automates the deployment processes, allowing for frequent updates to applications with minimal manual intervention. This approach helps in maintaining high development velocity, improving productivity, and ensuring that updates are delivered to users more rapidly and reliably.
- Automated Pipelines: Deployment pipelines in GitLab are meticulously configured to automate the entire deployment process. These pipelines handle everything from building applications and running tests to deploying containers. Automation in deployment not only speeds up the process but also significantly reduces the chances of human error.

Flexible Deployment Options:

- Hybrid Deployment Capability: The FALCON platform is designed to support both on-premises and cloud deployments, providing flexibility based on specific operational needs or security requirements. This dual capability ensures that the platform can be adapted to various environments, whether controlled internal networks or scalable cloud infrastructures.

- On-Premises Deployment Support: For on-premises deployments, technical support from partners is essential to ensure that the installation and setup processes are as straightforward as possible. The platform's design considers the need for adaptability to specific local conditions, which may require customized configurations or adjustments.

Container Management:

- Automated Container Deployment: All containers and tools within the FALCON ecosystem are deployed automatically through GitLab pipelines. This level of automation supports consistent deployments and facilitates rapid scaling and updates without disrupting the platform’s operations.
- Monitoring and Maintenance: Continuous monitoring and regular maintenance are integral to the deployment strategy, ensuring that the system remains efficient and secure. Monitoring tools integrated within the Kubernetes environment help detect and address potential issues before they impact the system’s performance or security.

The deployment strategy for the FALCON platform emphasizes automation, flexibility and robustness. Taking benefit of advanced technologies such as Kubernetes and adopting a CI/CD philosophy, FALCON ensures that its deployment processes will not only be efficient and error-free, but also adaptable to the changing needs of its users. This strategy ensures that the platform remains reliable and effective in its anti-corruption role, regardless of the deployment environment.

5. FALCON Secure Data Management and Knowledge Base

5.1 FALCON Secure Data Management

Assigned for Task 3.3 the creation of a secure data management and audit trail services will lead to the creation of a Secure Data Management and Audit Trail (SDMA). This module is envisioned to assume the technological capabilities of a secured data repository for evidence-based data types. Alongside the said repository is the capability to provide an audit trail that can be used as part of the data being used as digital evidence. The SMDA module is also set to facilitate the interaction of other modules that forms part of the FALCON system.

The Secure Data Management architecture has been crafted with a specific focus on meeting the security requirements of the FALCON platform. Drawing from the insights of past projects, such as AIDA[3] and integrating lessons from CREST[4], the architecture has been fine-tuned to provide optimal data storage, management, and protection within the system. SDMA offers a comprehensive data management capability, particularly on providing security measures for sensitive and large-scale data. In terms of software design, the module draws from the Model View Controller Service (MVCS) architecture. Its resilience has remained the definitive consideration to serve as the SDMA's foundation.

Here, a SDMA has been drawn to display the data management representation as a module in the overall platform of the project. It is anticipated to handle multimodal data types from a wide range of sources. Figure 10 provides an overview of its operational flow as a module interacting with the larger FALCON platform system.

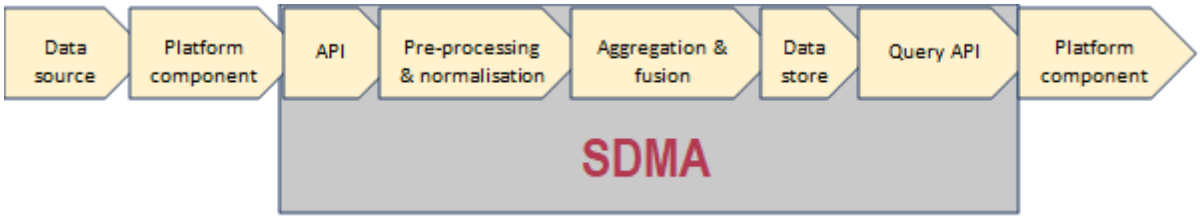

Figure 10. The Operational Framework of SDMA

The SDMA is both a Mongo and Janus graph-based data management system that employs data fusion to create links within the data models of the platform. This design allows for both advanced queries and relationship management to be performed via the Janus graph as well as document storage through the Mongo Database. The data models are pre-designed and developed by the consortium partners, as determined by the DOA. Nonetheless, these models are set to be gathered and placed into interaction with the

[3] https://www.project-aida.eu

[4] https://project-crest.eu

end-user by ensuring that the data collected and received by the module is normalized. As another module of the FALCON platform, it sets out a standardized JSON structure that the other data models are set to follow to ensure that the data received from all components of the system assumes a unified structure as information is stored and managed.

5.1.1 Data Source Model

The SDMA's data source model is anchored to several concepts – Artefacts, Entities and Links. These components of a data model will be responsible in defining the raw information collected and determine how such data will interact with the FALCON platform through the operational framework of SDMA.

For immediate reference, an Entity is best classified as something that is expected to produce data either internal within the system such as a user, vehicle or mission or external source such as websites, organisations, individuals, to name a few examples. An Artefact, on the other hand, is more relatable as documents, records, media, and posts produced by entities, and contains main content, exclusive to artefacts which contain readable text or file binary format.

Artefact and Entity, interchangeably used in this model, is defined as any data or concept collected or generated during an operation conducted by the end-user. As a media metadata stored in the SDMA, An Entity, or at times an Artefact, is used as a manifest and as a resource locator. In a general sense, however, it is used to provide a profile on the data collected, which is often translated as its metadata. Figure 11 below provides a quick snapshot of the recommended Entities that can be derived online.

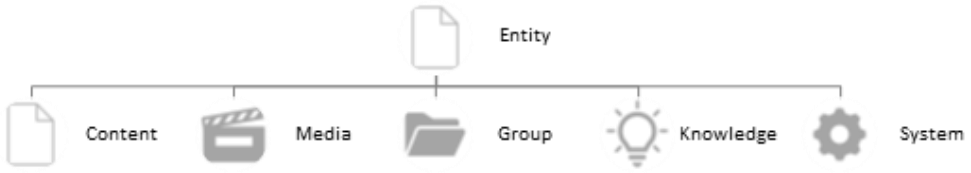

Figure 11. Recommended Virtual Entity Types

In addition to Figure 11, the table below provides an overview of an Entity's or Artefact's general anatomy, including the description of an Artefact in terms of its field and type.

Table 6. Artefact's General Anatomy

Field	Type	Description
domainId	Structured Text	The Domain Identifier for the entity (Explained in the domainId section)
Title	Plain text	Human readable name given to the entity

Raw	Plain text	The raw, unprocessed data as it was collected/generated
Content	Plain text	The plain text content of the entity
Source	Plain text	Field to indicate the originator of the data (website, social media, etc)
Raw Type	Plain text	The content mime-type
Attributes	Plain text dictionary	A dictionary of free fields for the storage of arbitrary metadata
Created	Datetime	The time of creation within the database
Updated	Datetime	Time when the data has been modified in the database
Verified	Boolean flag	Flag to indicate whether the signed hash matches the current data
Tags	List of structured text	List of metadata tags that are indexed in the database for filtering purposes

In terms of its role in the data model, an Entity and/or Artefact will be understood by the data model only if and when it is translated by a Domain Identifier. It serves as an Artefact's primary identification in the over-all data model and is expected to be uniquely dedicated to a single Artefact. To define such designation to its represented Artefact, a Domain Identifier can be constructed out of three distinct fields as follows:

Platform or Source Field – refers to where the raw data came from. An example could be information retrieved from a website like Twitter.

Type Field – refers to the definition of the specific format of the data. Examples could be an online post or a specific sensor output.

ID field – refers to the data's unique identifier and is set to be assigned to the said raw data since it has been generated as a content from the original data and all throughout, including while it is being processed in the SDMA.

The second component of SDMA's data source model is the Link model. It is used to define the relationships between two specific Entities, Artefacts, and/or a combination of either Entity and Artefact. It is expected to create the directed graph between the two intended artefacts in the SDMA, with the links being established through the said Entities' or Artefacts' domain identifiers. Figure 12 illustrates the role of the Link model.

Figure 12. Examples of two Links (arrowheads with text) in a Social Media Context; a post comment link and a comment to post link

In terms of data representation, structed textual data such as HTML, XML, JSONs whole structure can be stored within the "raw" field of an Entity or Artefact. Sometimes, this may be enough. However, to take full advantage of the SDMA's data source model and to be able to filter and search this data, a more accurate representation can be achieved by extracting fields from the structured data itself. These fields can then be stored within either the "content" field, in the case of the main text content of a web page, or using the "attributes" dictionary, for parts of a JSON data structure. In these cases, the raw field should still be used to hold the original data. Below is an example of an attributes dictionary and tags format in its key value. Other data formats, such as image resolution, tender price, and content language, these can be assumed as non-indexed formats and therefore will require a more strategic approach in defining data representations.

```
"attributes":{
  "Key1":["Value1","Value2"],
 "Key2":["Value3","Value4"]
}
"tags": ["Value1","Value2"]
```

Figure 13. Example of an Attributes Dictionary and Tags Format

5.1.2 Communication

Further complementing the above-described method, the FALCON system will be set to utilize Kafka. Utilizing this approach, which will directly interact with the SDMA module, Kafka will be used as the platform's orchestration instrument, providing dataflows between the various FALCON modules, using a chain reaction approach whenever state is changed within the system, modules are able to selectively choose which Kafka messages are relevant to them and should begin processing actions. An overview of this interaction can be seen in the figure below.

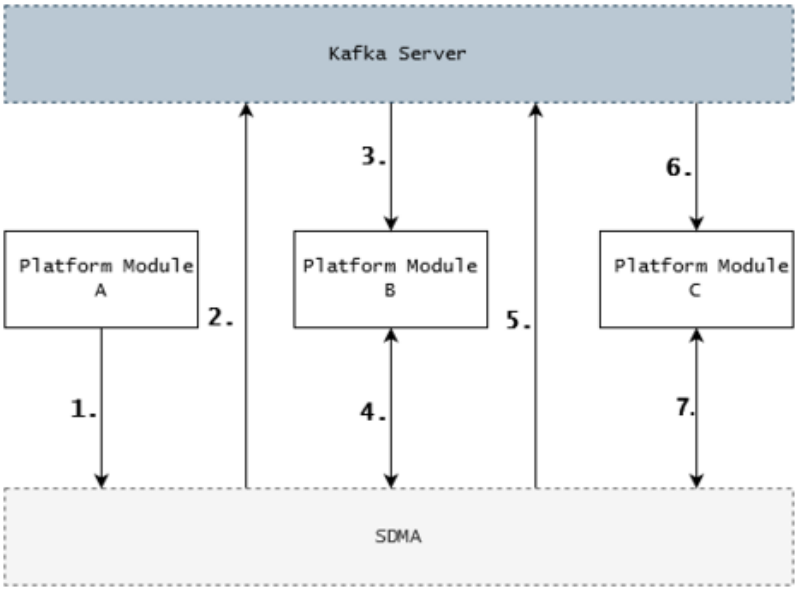

Figure 14. SDMA Interaction

Considering the above interaction, the SDMA pipeline can be considered to take place through the following steps:

1. An authenticated user using GUI to create a monitoring task in the online monitoring centre webpage, providing the web crawling and scraping parameters.
2. The task creation is sent to the GUI backend module for validation.
3. The task is sent to the SDMA REST API long with the user's SDMA access token to be stored as a pending crawling task.
4. The SDMA validates the users token using the public certificate.
5. The task is stored within the SDMA ready for the web scraping / crawling to begin.
6. Once the task begins, the collected data content will be streamed to SDMA as Artefacts, and this begins the pipeline

In a snapshot, Kafka serves a broadcasting channel for events within the SDMA, each time creation or mutation to data within SDMA, its internal Kafka producer pushes a message to the Kafka server, the message includes the DomainId of the artefact or entity, along with the project ID of the project it belongs to, and the type based on the domainId. As mentioned, RESTful API will provide reference for the FALCON system component's interactions since the intent of the SDMA persist to provide. It is intended to protect the underlying database query languages. Additionally, it is intended to prevent direct, and potentially malicious, database interactions, and thereby enhancing the secure data storage's security. While it operates through HTTP, provisions of security layers, such as TLS termination via a reverse proxy, is incorporated to further bolster the encryption of connections of modules in SDMA.

5.1.3 Functional Requirements

Table 7. SDMA Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-05	System must be able to analyse and cross-reference different media formats efficiently.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-09	Support concurrent access by multiple users without degradation in system performance.	Must-have
FUN-10	System must provide multi-language support to accommodate users in different regions.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have

5.1.4 Non-Functional Requirements

Table 8. SDMA Non-Functional Requirements

ID	Acceptance criteria	Ranking
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have
SEC-03	Enhanced security measures must be in place to protect sensitive data from unauthorized access.	Must-have
SEC-04	All data in transit and at rest must be encrypted; infrastructure must ensure data integrity and security.	Must-have
SEC-07	System must automatically save data at regular intervals or under specific conditions to prevent data loss.	Must-have
SEC-10	System must be robust and maintain operational capacity under various stress conditions.	Must-have
SEC-11	System must have comprehensive backup solutions to safeguard data integrity.	Must-have
SEC-13	Enhance security for user authentication to prevent unauthorized access.	Should-have

OPE-01	System must have failover capabilities and rapid recovery methods to ensure data integrity and availability.	Must-have
OPE-02	Interface must allow customization to meet the specific needs of different user types.	Must-have
OPE-04	Search functionality must be robust and capable of handling complex queries efficiently.	Must-have
OPE-08	Dashboard must update in real-time with full traceability and logging of all user actions and system responses.	Should-have
OPE-09	System must be designed to scale seamlessly as the amount of data and number of users grows.	Should-have
COM-01	System must support specified communication technologies to ensure wide accessibility and connectivity.	Must-have
COM-02	The system must facilitate secure and efficient communication between all relevant stakeholders.	Should-have
COM-03	System must ensure seamless integration with different IT environments and support various data formats.	Should-have
COM-04	Notifications must be timely and relevant, ensuring that all users are promptly informed of critical changes.	Should-have
COM-05	Collaboration tools must be secure, intuitive, and capable of supporting a multi-user environment.	Should-have
COM-06	The platform must be able to connect with various external systems and databases efficiently.	Should-have
COM-07	System should have mechanisms to alert users about system states, errors, or important notifications.	Should-have

5.1.5 Sequence Diagrams

As mentioned earlier, the inclusion of a MVCS architecture provides the store with a solid foundation. By introducing a service layer, the MVCS design separates the business logic from the controllers, enhancing the store's resilience. This separation means that shifts in business operations or adjustments to the system's underlying structure have no detrimental impact on the API's security or functionality. In essence, each component within the MVCS design has a well-defined role, ensuring clarity and minimizing potential vulnerabilities, as illustrated in the architecture diagram in Figure 15 below.

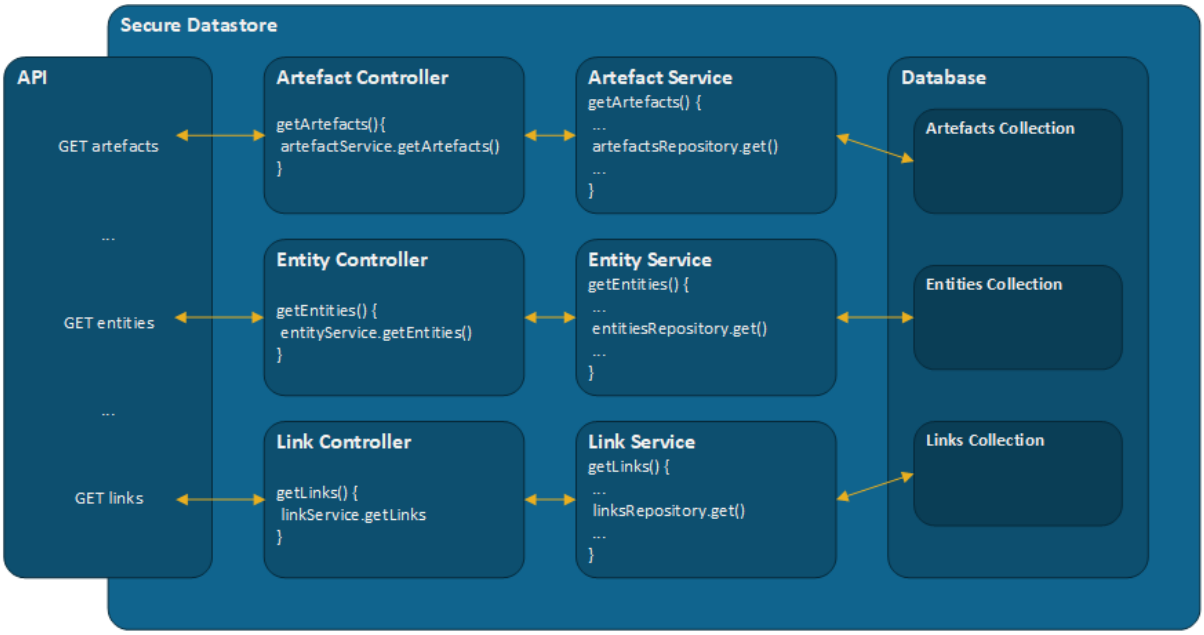

Figure 15. SDMA Architecture Diagram

5.1.5.1 *Audit trail for digital evidence*

During its operation, the FALCON system will handle data from many sources, with the expectation that this data can be used in a court of law to support an investigation. Therefore, a component that defines the flow of data within the system and audits each action within the system is important to retain the chain of custody on all information handled by the platform.

To ensure that the audit trail is both secure and tamper resilient, a methodology employing blockchain technologies including cryptographic hashing has been devised. This process builds upon the fundamental operations of a blockchain, commonly seen as public distributed ledgers for cryptocurrencies. These large public blockchains provide several guarantees that are useful to an audit trail, however, have some features that are not useful in this use case. For example, proof of work which aims to solve the problem of having no single source of truth within the system, instead block creators (miners) prove they have significant stake by performing computationally expensive calculations. In a closed Audit system without untrusted public access this is unnecessary, as there is a degree of trust between system components.

To begin at a functional level, each change to data or data access within the FALCON system is logged, the secure data store will provide these changes to the audit component via a RESTful API. This Component will be entirely separate from the Secure Data Store, to increase both resilience and security, and to also delegate the responsibility to a dedicated component, reducing the complexity of the secure data store.

5.1.5.2 *Audit Item Data Model*

The main data model of the audit trail, Audit items represent each audit action within the system. Audit items are designed to be generic and can represent many different types of audit actions ranging from data access to system logon events, data writes, and modifications.

```
{
  Type: Access
  application: <project>
  bucket: default
  identifier: null
  created: 2022-06-15T15:00:00Z
  data:
  {
    url: /api/projects
    query: null
  }
  auth:
  {
    <User/Module>
  }
  Hash: <sha256>
}
```

Figure 16. Audit Item

The Audit Trail data model of the SDMA in its basic form assumes the top-level API route of /api/ providing buckets to collect audits in difference contexts; default buckets include registration of user/modules with the SDMA, and a bucket to register the creation of new project from within the SDMA; all other audit item API requests support project level buckets by providing the project ID in the submission of an audit item. This allows audit items to have a bucket per project, each containing a blockchain. The audit items are added to a blockchain-like technology with intent to provide resistance to potential data tampering and assume data integrity. Its business logic and control flow are also illustrated in the diagram below.

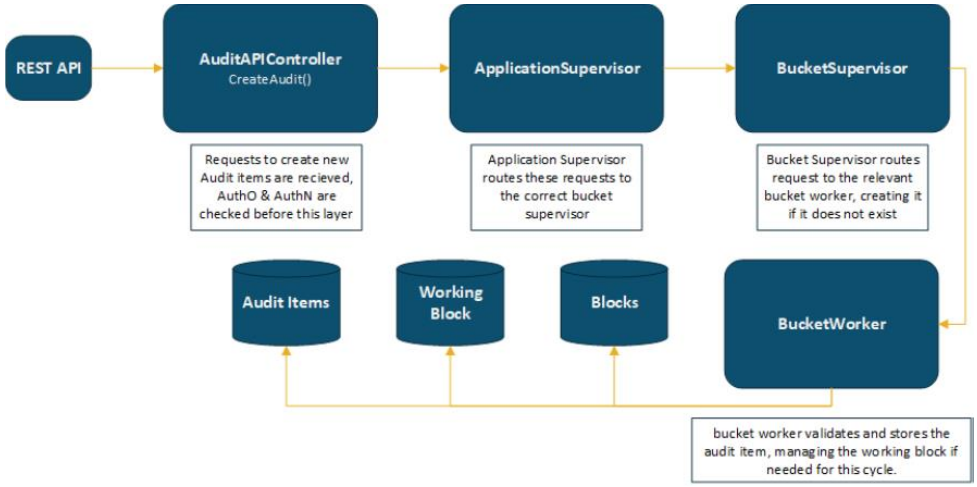

Figure 17. SDMA Data Control Flow

5.2 FALCON Knowledge Base

The FALCON Knowledge Base (KB) is a type of database where the conclusions / analysis results from the various tools of the FALCON platform can be stored / retrieved in a structured format through a dedicated Application Programming Interface (API). The FALCON KB consists of the following four main components which are shown in the UML component diagram of Figure 18:

- The FALCON Ontology
- The Triplestore Database (TDB)
- The KB API
- The KB User Interface (UI)

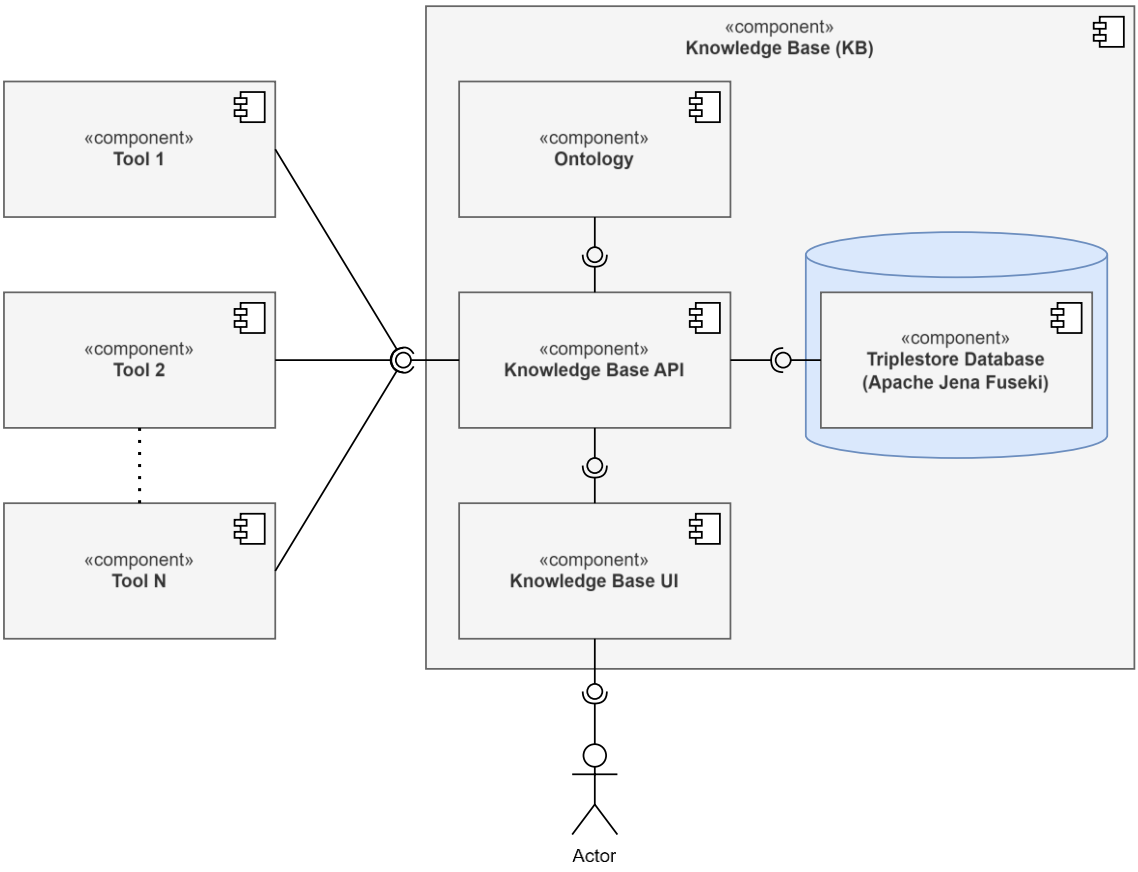

Figure 18. Interfaces and Components of the FALCON Knowledge Base

The TDB is used to store all the relevant high-level information gathered from the FALCON tools in the form of Resource Description Framework (RDF) triples consisting of <subject> <predicate> <object>, e.g. information like <PersonA> <collaborates with> <PersonB>. For the TDB Apache Jena Fuseki is used which is a SPARQL server and RDF database that is part of the Apache Jena framework. It allows for storing, querying, and managing RDF data using the SPARQL 1.1 Query and Update standard [5].

[5] https://jena.apache.org/

The domain-specific ontology defines the FALCON Common Representational Model (CRM) which is a formal representation of a set of concepts and the relationships between those concepts within the FALCON domain. Its primary use is to standardize and share knowledge across different systems and applications within the domain. The ontology itself is implemented using a Semantic Web language, the W3C Web Ontology Language (OWL)[6].

When combined, the TDB and FALCON CRM provide a powerful framework for a common understanding of the semantics of the data in order to support decision-making processes which allows for semantic queries, complex relationships and attributes to be inferred and queried in a way that traditional databases cannot.

The KB API provides a Representational State Transfer (REST) HTTP interface that encapsulates the Fuseki interface which is used for accessing the data in the FALCON KB while maintaining the (referential) integrity. This means that only triples that correspond to the classes and relationships defined in the ontology can be instantiated. Furthermore, it ensures that only existing or newly created instances can be referenced or, if instances are removed from the KB, that all references to them are removed.

Finally, the KB UI makes it easier for an end-user to interact with the data in the KB, e.g. to search for, create, query, update, and delete triples and to explore the FALCON CRM, offering different views on the data by visualizing the knowledge graph and providing a tabular view. Visual analytics by exploring the knowledge graph can significantly enhance the efficiency and effectiveness of investigative processes by providing a clear and intuitive way to navigate and analyse complex datasets. A knowledge graph visualization

- helps in identifying patterns and correlations among data points that might not be obvious in traditional data analysis;
- facilitates the understanding of complex networks, such as criminal networks or financial transactions, by visually displaying the connections and how they interact;
- provides a comprehensive overview that aids investigators in making informed decisions based on the visual insights extracted from the data.

5.2.1 Functional Requirements

Table 9. FALCON Knowledge Base – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-01	The documentation / user manual of the KB, including the ontology and UI, must be clear and easy to use.	Must-have

[6] https://www.w3.org/OWL/

FUN-02	The UI of the KB, must be easy to use and must provide different views on the data (graph-based / tabular view).	Must-have
FUN-05	It must be possible to cross-match multi-media content by interrelating information in the KB to raw data.	Must-have
FUN-09	It must be possible that users can work in parallel on the case.	Must-have
FUN-10	The KB must support different localizations in the UI.	Must-have
FUN-11	It must be possible that the KB can be operated as a standalone application.	Must-have
FUN-12	The KB must have a high-performance database in the backend.	Must-have
FUN-14	The system requirements for an end-user device should be modest by using a web-interface that can be accessed using a standard web-browser.	Should-have

5.2.2 Non-Functional Requirements

Table 10. FALCON Knowledge Base – Non-functional Requirements

ID	Acceptance criteria	Ranking
SEC-02	The KB must log all data access events to enable audit and traceability.	Must-have
SEC-04	The KB must be hosted locally and the traffic between the KB and the client must be encrypted by using a reverse-proxy.	Must-have
SEC-07	The data in the KB must be periodically saved in order to prevent data-loss.	Must-have
SEC-11	Data backups of the data in the KB must be possible.	Must-have
OPE-01	Data backups of the data in the KB must be possible. It must be possible to restore backups.	Must-have
OPE-03	The analysis results in the KB must be visualized.	Must-have
OPE-04	The information in the knowledge graph must be searchable (keyword search / refined search).	Must-have

5.2.3 Sequence Diagrams

The sequence diagrams in this section show the interactions between the different components of the FALCON KB. Please note that the user and the KB UI can be replaced by a FALCON tool which communicates with the KB API directly.

5.2.3.1 *FALCON Knowledge Base – Queries*

The KB supports the SPARQL Protocol and RDF Query Language (SPARQL)[7] for accessing the data stored in the TDB. SPARQL supports complex querying capabilities such as filters, aggregations, and sub-queries, providing robust tools for data analysis and manipulation. The following sequence diagram shows the interactions between the components of the KB when the user searches for information in the KB using a suitable query.

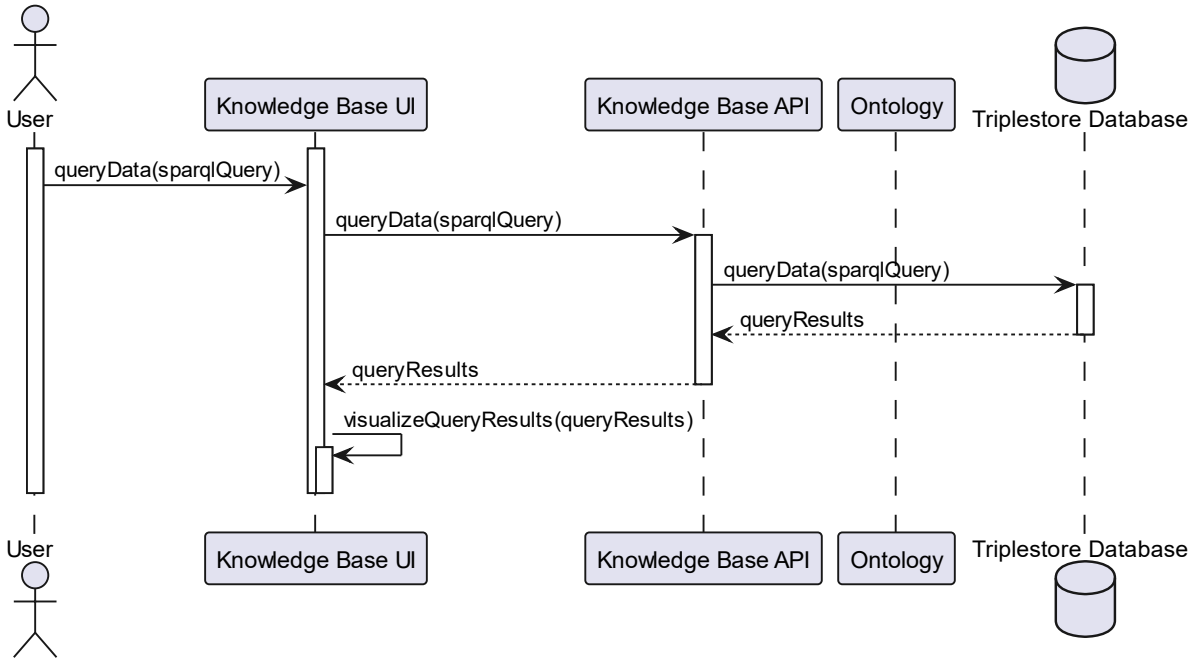

Figure 19. UML Sequence Diagram for Knowledge Base Queries

A SPARQL query can be formulated by the user in the KB UI. These are directly sent through the KB API to the SPARQL endpoint of the TDB where it is then evaluated. The results are sent back to the KB UI where they are presented to the user in a proper way.

5.2.3.2 *FALCON Knowledge Base – Data Ingestion*

For ingesting new content into the KB, different triple formats like RDF/XML (Resource Description Framework / Extensible Markup Language) and Turtle (Terse RDF Triple Language) are supported as well as update operations using SPARQL updates[8]. The interactions for adding new data, as shown in Figure 20, are more complex, because various checks need to be carried out in order to maintain the referential integrity of the content in the KB.

[7] https://www.w3.org/TR/rdf-sparql-query/

[8] https://www.w3.org/TR/sparql11-update/

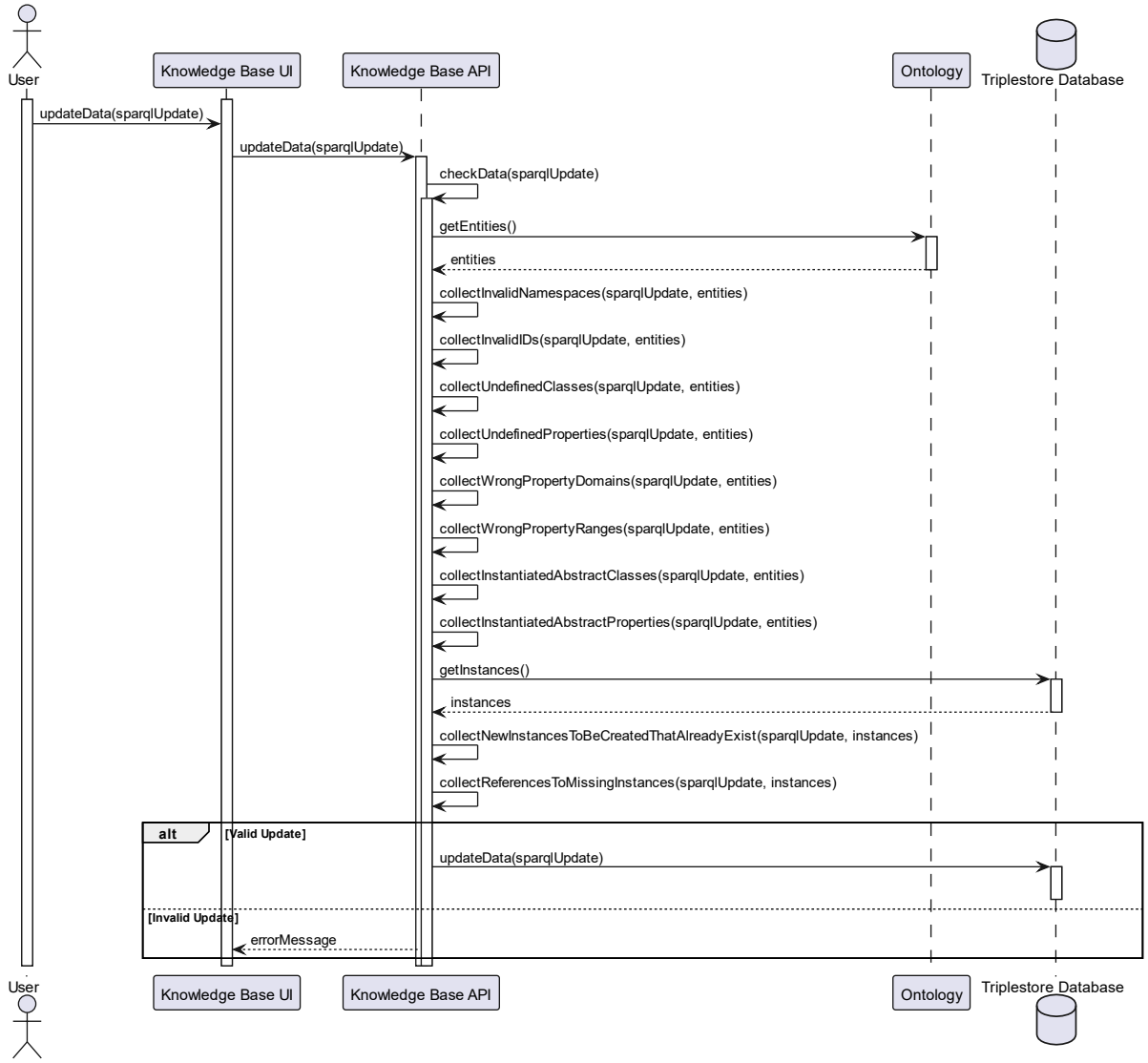

Figure 20. UML Sequence Diagram for Knowledge Base Updates (insert operation)

Ingesting data can be performed directly in the KB UI by formulating a suitable SPARQL update or by adding triples. This operation is sent to the KB API where it is evaluated and where various checks are performed before the data is finally persisted into the TDB:

- First, the ontology is queried to get all the relevant classes and relations that are mentioned
- The data to be inserted is then checked for potential issues, e.g. if the triples:
    - Have incorrect namespaces
    - Have invalid ID's (e.g. the ID of an instance is named after a class in the ontology)
    - Contain (instances) of classes or properties that are abstract or not defined in the ontology

    - Are using incorrect domains and ranges depending on the corresponding property
- The instances relevant for the update are then queried from the TDB in order to collect issues with:
    - Already existing instances in the KB which may be overwritten
    - Data to be inserted containing references to non-existent instances
- If there are no issues, the data is ingested into the TDB, otherwise a corresponding error message with further details will be sent back

5.2.3.3 *FALCON Knowledge Base – Deletions*

Like when ingesting data, delete operations can be performed using triples in a specific format or using a SPARQL update. This process involves the following interactions:

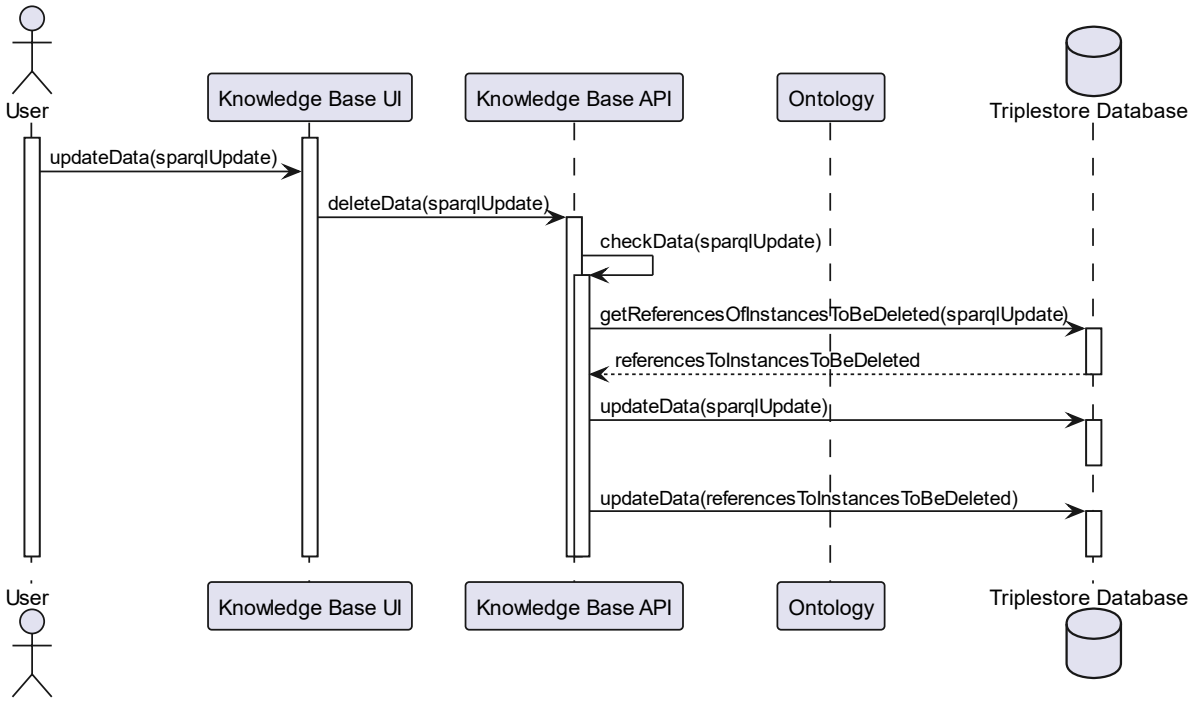

Figure 21. UML Sequence Diagram for Knowledge Base Updates (delete operation)

The update is sent to the KB API, which collects all references to instances to be deleted. This ensures, that another instance does not reference an instance has already been deleted. Finally, all instances mentioned in the update are removed from the TDB along with their references.

5.2.3.4 *FALCON Knowledge Base – Knowledge Graph Visualization*

The knowledge graph within the TDB can be used for visualizing the triples, where the nodes are the subjects and objects connected by their corresponding property, representing the edge between two nodes.

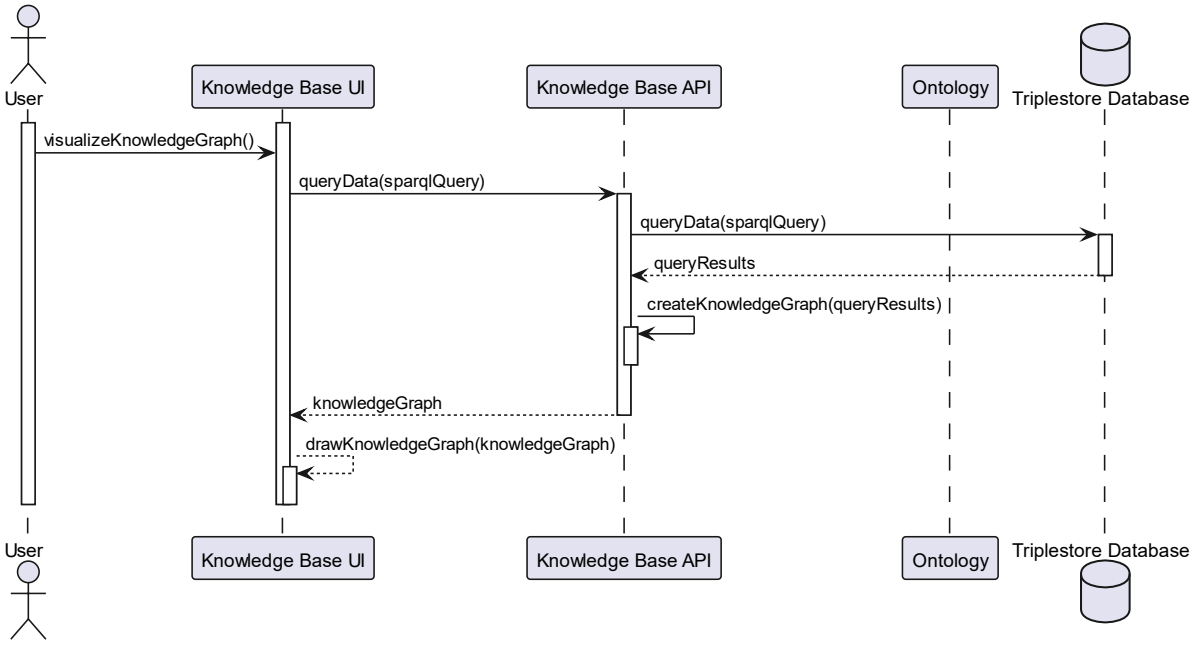

Figure 22. Knowledge Graph Visualization

The knowledge graph visualization involves a suitable query as described in Section 5.2.3.1. The query results are then converted to a special JSON graph format that is sent back to the KB UI where it is visualized using Cytoscape.js [1]. Cytoscape.js is a graph theory library used in various domains (e.g. bioinformatics) to analyse and visualize graphs or (social) networks.

6. Trustworthy AI

In the FALCON project, the trustworthiness and security of AI models are paramount to ensure that all AI models developed are deployed securely. To this end, T3.4 focuses on studying the trustworthiness and security of AI models. This task involves sourcing various approaches to enhance the robustness of AI models, including insights from the research community, initiatives, and regulatory frameworks.

The primary goal is to establish this initial study as a baseline for trustworthiness and security in the FALCON project's developments. Furthermore, adherence to current European regulations is mandatory, ensuring that the baseline is not just a guideline but a requirement for AI model development.

On May 8, 2024, T3.4 hosted a webinar within the FALCON consortium to present their findings on AI trustworthiness and security. The purpose of this workshop was to introduce all consortium partners to the legal, ethical, and security aspects of AI models. The webinar aimed to raise awareness about the cybersecurity threats inherent in AI models and provided detailed information to ensure compliance with the established baseline in line with European AI regulations.

Moving forward, the FALCON project will continue to monitor emerging regulations and research in AI security. This ongoing effort will ensure that the study is continuously updated, maintaining its relevance and effectiveness in safeguarding AI developments. Moreover, this continuously updating will be share with the consortium of FALCON via other two webinars during the project.

6.1 Motivation

The global surge in AI investments by industries and governments underscores the transformative potential of Artificial Intelligence in unlocking new business opportunities and enhancing existing ones. However, the inherent vulnerabilities of AI systems, if left unprotected, expose them to threats that can undermine their functionality and integrity. Effective AI security is crucial to safeguard these technologies against potential risks and misuse, ensuring their safety and reliability. Current regulations and protective measures are insufficient, leaving AI systems vulnerable to exploitation by malicious actors. The risk of misuse is significant, with inadequate protection enabling harmful manipulation of AI models. As awareness of these risks grows, organizations are increasingly prioritizing AI security, driven by expert recommendations for robust defence plans and better regulatory adherence. There is a pressing demand for clear guidance, effective tools, and straightforward methods to secure AI systems and mitigate potential risks. Businesses are embracing the importance of integrating safety measures from the beginning, adopting a 'security by design' approach that embeds essential safety features into AI systems from their

inception. This proactive strategy not only protects investments but also builds customer trust and satisfaction by ensuring the reliability and safety of AI technologies. Addressing these security challenges is essential for realizing the full potential of AI while maintaining ethical standards and protecting against malicious use.

AI security is a critical concern due to the inherent vulnerabilities of AI systems to traditional cybersecurity threats. Protecting these systems requires comprehensive security measures across their entire architecture. As AI technologies evolve, the need for enhanced defences becomes paramount to address emerging security risks such as model extraction, data leakage, system malfunctions caused by manipulated inputs, and the evasion of intended functionality. Additionally, AI introduces new attack vectors, including data manipulation, training techniques, compromised framework libraries, and vulnerabilities within the AI systems themselves. Overcoming these challenges necessitates innovative system design and robust defence strategies. The field of AI security, or AI cybersecurity, is rapidly emerging to focus on identifying, detecting, preventing, and responding to malicious attacks targeting AI systems. This discipline encompasses the identification of AI model vulnerabilities, understanding the capabilities of attackers, assessing the potential consequences of attacks, and building resilient AI-enabled systems. The primary emphasis is on cybersecurity for AI, highlighting protective measures specifically for AI systems rather than leveraging AI to solve broader cybersecurity issues. This focus ensures that AI technologies are safeguarded against threats, maintaining their integrity and reliability.

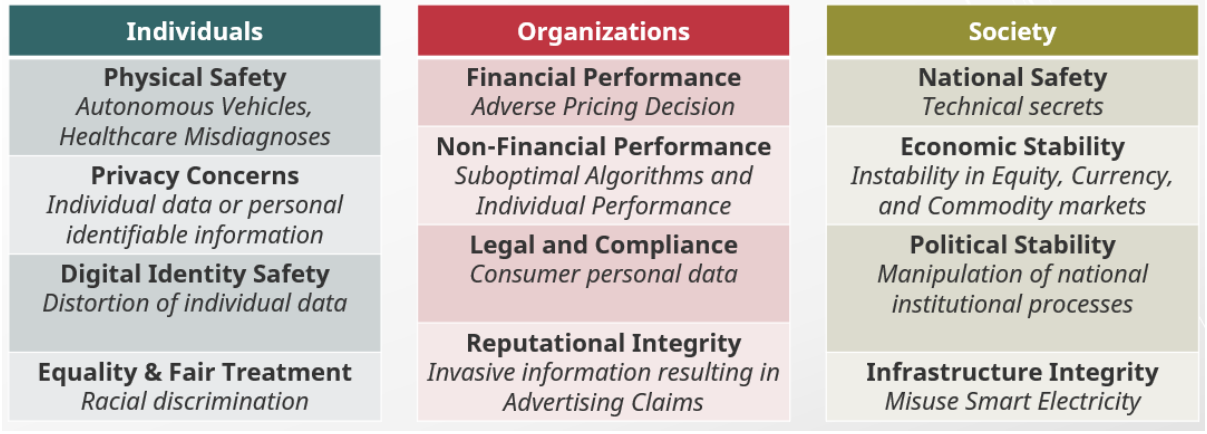

Figure 23. Potential Impacts of AI and Data Misuse

6.2 Overview of AI Security

AI security presents a formidable challenge to organizations worldwide due to the complexity and interconnectedness of AI systems, which create numerous avenues for exploitation. The dynamic cybersecurity landscape, evolving from traditional threats to those specific to AI, demands proactive measures to protect sensitive data and infrastructure. To effectively mitigate AI security threats, it is crucial to have a deep

understanding of the unique vulnerabilities inherent in AI systems and to develop robust defence strategies. This involves exploring various dimensions of AI security threats, such as algorithm manipulation and unauthorized data access, to better prepare for emerging challenges. By comprehensively understanding the nature and potential impact of AI security threats, organizations can more effectively protect their AI-driven initiatives and mitigate associated risks. Proactive measures, combined with collaboration across industries and sectors, are essential for navigating AI security threats and safeguarding digital assets in an increasingly interconnected world.

6.2.1 Research Community

The research community in AI security has seen significant developments since the origins of Adversarial Machine Learning (AML) research in 2004. It began with the discovery that carefully crafted spam emails could deceive linear classifiers used in spam filters, highlighting the potential for adversarial manipulation. Following this, numerous studies have been conducted to further explore this threat and propose various mitigation measures. The evolution of AML research is comprehensively documented in the 2018 paper "Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning," which provides an overview of the field's progression over a decade.

Around 2013, the focus of AML research began to shift towards the security of Deep Neural Networks (DNNs), particularly concerning evasion attacks, where adversaries craft inputs to fool the models. This shift was driven by the increasing deployment of DNNs in various critical applications, necessitating enhanced security measures to protect against sophisticated attacks.

In recent years, the academic interest in AI security has surged, reflected by an exponential growth in the number of publications related to the field. This burgeoning body of research underscores the importance and urgency of addressing AI security challenges, as well as the collaborative efforts within the research community to develop robust defences against evolving threats.

Table 11. AI Security Publications

Attacks	Information	References
Evasion	Adversarial attacks have been highly effective, leading to a main defence strategy centred on adversarial training to improve robustness. However, there's often a trade-off between model performance and robustness, especially noticeable in neural networks. Recent defence efforts focus on refining adversarial training methods, understanding the balance between utility and robustness, and creating models with proven robustness. Yet, achieving this remains challenging due to scalability issues with real-world datasets and large neural network sizes.	[2],[3],[4],[5]

Model Extraction	In recent years, researchers have learned more about how to extract models, thanks to studies in different areas. These studies cover things like computer vision, Recurrent Neural Networks (RNNs), and tabular data. They often use a method where they predict queries to extract the model. But now, with better defences, just looking at the predictions isn't enough to get the model. Another way to extract a model, especially for Edge AI devices, is to use information about how the model leaks out or data from side channels like timing, power, and memory access.	[6],[7]
Inference	Machine learning models can be attacked to reveal private training data, either through legitimate or carefully crafted queries aimed at exposing confidential model details. These attacks occur at both algorithmic and physical levels, exploiting side-channel emissions. Limiting the number of queries on a model is a straightforward but not highly effective method to mitigate these attacks, especially since it's impractical for most real-world machine learning applications.	[8],[9],[10]
Poisoning	Dataset poisoning and model backdoor insertion are mainly studied in computer vision. Recent attacks in this area are effective, often needing only a small amount of corrupted training data to activate the backdoor during model deployment. These attacks assume accessible and tamperable training data. Current defence strategies focus on identifying tainted data by analysing its statistical properties compared to legitimate data.	[11],[12]

6.2.2 Community: Consortiums & Initiatives

In response to the growing concern over AI security, several initiatives and consortiums within the cybersecurity community have emerged to analyse the associated challenges and provide guidelines and tools to understand and mitigate potential threats to AI models.

One notable initiative is MITRE ATLAS[9] (Adversarial Threat Landscape for Artificial-Intelligence Systems). ATLAS is a comprehensive knowledge base detailing adversary tactics, techniques, and case studies for ML systems. This resource is grounded in real-world observations, demonstrations from ML red teams and security groups, and insights from academic research. Modelled after the MITRE ATT&CK® framework, ATLAS offers complementary tactics and techniques specific to AI, enhancing the existing ATT&CK framework.

The AI Security Alliance[10] is another key player in this space, focusing on educating stakeholders about how AI can bolster an organization's security posture and

[9] https://atlas.mitre.org/

[10] https://aisecurityalliance.org/

developing best practices around AI security. This consortium seeks to define and disseminate knowledge on integrating AI effectively and securely within organizational frameworks.

The Foundation for Best Practices in Machine Learning[11] advocates for ethical and responsible machine learning by promoting open-source best practices. This initiative aims to assist data scientists, governance experts, managers, and other ML professionals in implementing ethical and responsible AI systems through its free, open-source guidelines. Their efforts are centred on technical and organizational best practices for machine learning.

Additionally, the Building a Secure Machine Learning[12] (BIML) initiative focuses on three main areas: constructing a taxonomy of known attacks on ML systems, exploring hypotheses related to representation and ML risk, and performing architectural risk analyses (or threat modelling) of ML systems in general.

Lastly, the Guaranteeing AI Robustness against Deception[13] (GARD) program seeks to establish theoretical foundations for ML systems to identify vulnerabilities, characterize properties that enhance system robustness, and foster the development of effective defences. Through these endeavours, GARD aims to build more resilient AI systems capable of withstanding adversarial attacks.

Together, these initiatives represent a concerted effort within the AI and cybersecurity communities to address the complex challenges of AI security, promoting the development of robust, ethical, and secure AI systems.

6.2.3 Regulation, Standards & Guidelines

The landscape of AI security is increasingly guided by a series of regulations, standards, and guidelines that aim to address the complexities and vulnerabilities associated with AI systems. Various organizations and frameworks play crucial roles in shaping these efforts, providing comprehensive strategies and recommendations for ensuring AI security.

The ENISA Threat Landscape report, developed by the European Union Agency for Cybersecurity (ENISA) with the support of the Ad-Hoc Working Group on Artificial Intelligence Cybersecurity, offers a detailed mapping of the AI cybersecurity ecosystem. This report lays the groundwork for future cybersecurity policy initiatives and technical guidelines, emphasizing critical challenges, especially those related to the AI supply chain. It highlights the necessity for an EU ecosystem dedicated to secure and

[11] https://www.fbpml.org/

[12] https://berryvilleiml.com/

[13] https://www.darpa.mil/program/guaranteeing-ai-robustness-against-deception

trustworthy AI, which prioritizes cybersecurity and data protection while promoting innovation, capacity-building, awareness, and research and development initiatives.

The draft EU AI Act represents another significant regulatory framework, proposing harmonized rules for the utilization of AI systems within the European Union. This legal framework categorizes AI systems based on their associated risks: systems with 'unacceptable' risks are prohibited, 'high-risk' systems are subject to stringent requirements to access the EU market, and 'limited risk' systems face minimal transparency obligations. Noncompliance can lead to substantial fines, up to €30 million or 6% of the total worldwide annual turnover of the offending entity.

The International Organization for Standardization (ISO) provides documents on the trustworthiness of AI systems. These documents cover approaches to establishing trust through transparency, explainability, and controllability; identifying engineering pitfalls and associated threats; and assessing and achieving AI system attributes such as availability, resiliency, reliability, accuracy, safety, security, and privacy. Although these documents do not specify levels of trustworthiness, they offer a comprehensive survey of methods to enhance trust in AI technologies.

The Securing Artificial Intelligence (SAI) initiative introduces a Securing AI Threat Ontology to align terminology, focuses on data issues and risks in AI training, provides mitigation strategies to address AI threats, and emphasizes the role of hardware in AI security. This initiative offers guidance on security testing of AI systems and addresses the data supply chain's vulnerabilities.

The IEEE has developed standards covering various aspects of AI security, including multi-party computation, Federated Learning, and shared Machine Learning. The IEEE 7000 series specifically addresses AI ethics, providing a framework for ethical AI system development.

The NIST AI Risk Management Framework tackles the challenges of managing AI-related risks. It provides guidelines for how AI actors can better identify, assess, prioritize, respond to, and communicate about these risks, improving overall AI risk management practices.

Lastly, the BSI AIC4 (AI Cloud Service Compliance Criteria Catalogue) from the Federal Office for Information Security (BSI) focuses on evaluating the trustworthiness of AI-based services developed and operated in the cloud. This catalogue covers a range of topics and provides criteria for assessing the security and robustness of AI systems, including the evaluation of risks from malicious attacks and the effectiveness of defence measures.

Together, these regulations, standards, and guidelines form a robust framework aimed at enhancing the security, trustworthiness, and ethical deployment of AI systems, ensuring that AI technologies are developed and used responsibly and securely.

7. Functional Description of the FALCON Tools

The FALCON project adopts a user-centric methodology combined with a technology-driven perspective. In other words, the consortium's technology partners introduce prototypes at different stages of maturity into the FALCON ecosystem. Based on user needs, these prototypes are developed, customized, and integrated to provide the functionality needed to improve the management of anti-corruption actions. This is a cyclical and iterative process, which emphasizes the identification and definition of user requirements. This procedure has been applied in Document D3.1, which addresses the functional, security, operational and user communication requirements that serve as the cornerstone of this Section.

As a result, our technology partners have successfully converted user needs into system requirements. These system requirements comprehensively cater to all aspects of FALCON platform, considering every vertical and horizontal integration point.

The system requirements' classifications encompass all elements of the FALCON framework along with Horizontal-Generic needs, with the latter pertaining to multiple or even every tool.

In this deliverable the user requirements are named and listed. You can examine each of them in more depth in D3.1 titled "Use Cases and Requirements".

Therefore, in this Section, the system requirements for FALCON are showcased, using distinct subsections and separate tables for each tool.

Each system requirement has:

- **ID:** a unique identifier. (From D3.1)
- **Acceptance Criteria:** specific standards or conditions that a product or feature must meet in order to be accepted by the user.
- **Ranking:** The priority of the related end-user requirement.
    - Must-have
    - Should-have
    - Could-have
    - Won’t-have

An additional column will be added in next versions of FALCON Framework Architecture documents where the status of the requirement will be updated according to the development/integration activities.

- **Status: status of the system requirement**
    - Not started
    - Under research
    - Under development
    - Fulfilled

7.1 FALCON Communication Broker

7.1.1 Description

The FALCON Communication Broker is a pivotal tool within the FALCON platform, designed to manage and facilitate seamless communication between various components and external systems. It acts as an intermediary, ensuring that data and messages are correctly routed and exchanged in a timely and secure manner. The Communication Broker supports various communication protocols and formats, enabling interoperability between different systems and enhancing the overall efficiency of the FALCON framework.

By integrating the FALCON Communication Broker, the platform can handle real-time data streams, alerts, and notifications effectively. This tool ensures that critical information is promptly distributed to the relevant parties, supporting collaborative efforts among investigators and other stakeholders. Its robust communication infrastructure is vital for maintaining the flow of information across the platform, ensuring that all components work in harmony to achieve the project's objectives.

7.1.2 Functional Requirements

Table 12. Falcon Communication Broker – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-09	Users can work simultaneously on cases with real-time updates facilitated by the Communication Broker.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have

7.1.3 Non-Functional Requirements

Table 13. Falcon Communication Broker – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	System must have failover capabilities and rapid recovery methods to ensure data integrity and availability.	Must-have

	System must enforce user access based on predefined roles, ensuring secure access control.	
OPE-02	Interface must allow customization to meet the specific needs of different user types. System must log all data access events to enable audit and traceability.	Must-have
OPE-09	System must be designed to scale seamlessly as the amount of data and number of users grows.	Should-have
COM-01	Compatible with at least one of the communication technologies (fiber, 4G/5G, TETRA, satellite).	Must-have
COM-02	Ensures effective communication between all persons or LEAs involved in a specific case.	Should-have
COM-03	Integrates with existing IT infrastructure, enabling import/export of data in different formats.	Should-have
COM-04	Notification features work correctly, providing real-time alerts on critical updates or changes.	Should-have
COM-06	Supports connectivity to different systems and databases without issues.	Should-have
COM-07	Implements effective warnings and alarm systems for immediate attention to critical situations.	Should-have
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have
SEC-03	Enhanced security measures must be in place to protect sensitive data from unauthorized access.	Must-have
SEC-04	All data in transit and at rest must be encrypted; the infrastructure must ensure data integrity and security.	Must-have

7.1.4 Sequence Diagram

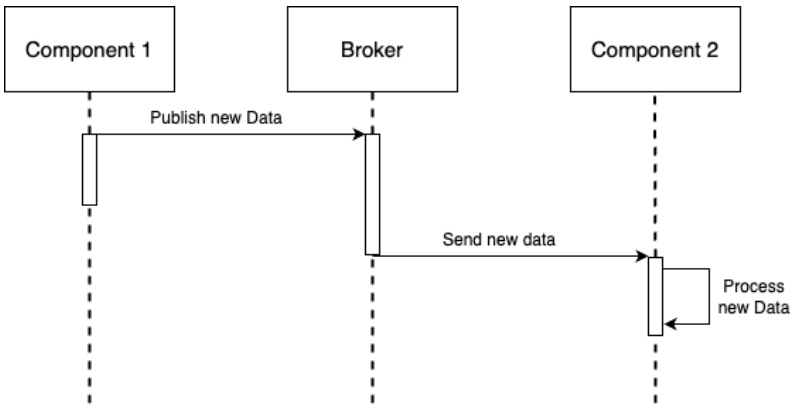

Figure 24. FALCON Communication Broker – Sequence Diagram

7.2 Streamsets

7.2.1 Description

StreamSets is a powerful data integration tool that facilitates the collection, transformation and movement of data between various systems. Within the FALCON platform, StreamSets is a key component to ensure efficient and seamless data integration from multiple sources. Its capabilities enable real-time data ingestion, processing and distribution, making it an essential tool for managing the large volumes of data for which FALCON is designed.

StreamSets enables automated ingestion and transformation of data from a variety of external sources, such as multimedia content and other structured and unstructured data sets. This ensures that data is accurately and quickly ingested into the FALCON framework, facilitating subsequent analysis and processing tasks. The tool's ability to handle complex data pipelines and integrate seamlessly with other tools makes it indispensable for maintaining data integrity and flow within the platform. With StreamSets, FALCON ensures efficient management of data from multiple sources, improving the overall functionality and reliability of the platform.

7.2.2 Functional Requirements

Table 14. Streamsets – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	StreamSets successfully connects to all existing databases as per the configuration and access permissions.	Must-have
FUN-07	Data from external sources is consistently and accurately ingested into the platform.	Must-have
FUN-08	StreamSets can handle and process large volumes of "Big Data" without performance degradation.	Must-have
FUN-11	The system continues to operate independently from current infrastructure, even during high data loads.	Must-have

7.2.3 Non-Functional Requirements

Table 15. Streamsets – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-02	User-friendly interface and customizable dashboards are available and functioning as intended.	Must-have
OPE-04	Keyword and refined search functions return accurate and relevant results during operation.	Must-have
COM-03	StreamSets integrates smoothly with existing IT infrastructure, supporting seamless data exchange.	Should-have

COM-06	The tool supports connectivity to various systems and databases without any compatibility issues.	Should-have
SEC-07	Data is periodically saved, and integrity checks confirm no data loss occurs during cyber-attack simulations.	Must-have
SEC-11	Regular backups are completed without errors, ensuring data can be restored as needed.	Must-have

7.2.4 Sequence Diagram

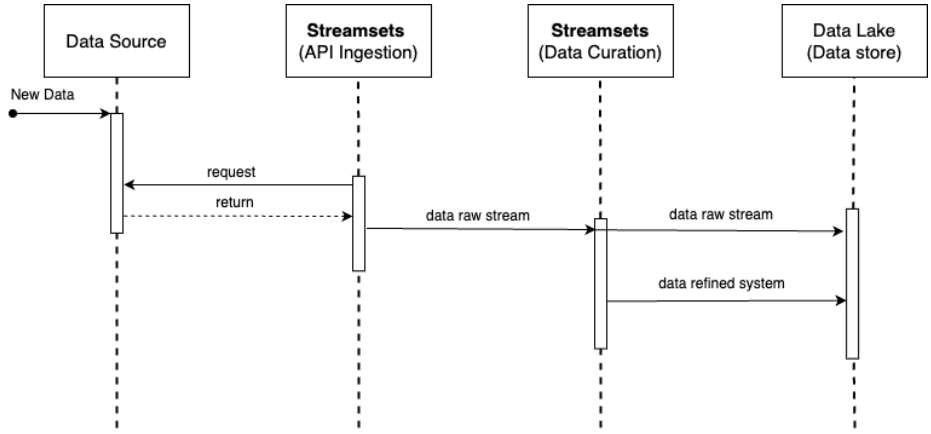

Figure 25. Streamsets Sequence Diagram

7.3 Apache Nifi

7.3.1 Description

Apache NiFi is an integrated data logistics platform for automating the movement of data between disparate systems. Within the FALCON platform, Apache NiFi plays a critical role in orchestrating the flow of data from various sources, ensuring efficient, secure, and scalable data handling. NiFi's powerful data routing capabilities enable it to collect, transform, and distribute data in real-time, making it an essential tool for the dynamic data needs of FALCON.

NiFi supports the FALCON platform by providing a visual interface for designing data flows, which simplifies the management of complex data pipelines. Its capabilities include data ingestion from multiple sources, real-time analytics, and data transformation. By leveraging NiFi, the FALCON platform can ensure that data is accurately captured, processed, and made available for analysis and decision-making. Additionally, NiFi's robust security features ensure that data is handled securely, complying with various data protection standards.

7.3.2 Functional Requirements

Table 16. Apache Nifi – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	NiFi successfully connects to and integrates with all existing databases and data sources.	Must-have
FUN-07	Data from external sources is consistently and accurately ingested into the platform.	Must-have
FUN-08	NiFi can handle and process large volumes of “Big Data” without performance degradation.	Must-have
FUN-11	The system operates independently from current infrastructure, with NiFi managing data flows.	Must-have

7.3.3 Non-Functional Requirements

Table 17. Apache Nifi – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-02	NiFi includes a user-friendly interface with customizable dashboards for monitoring data flows.	Must-have
OPE-04	Keyword and refined search functions return accurate and relevant results during operation.	Must-have
COM-03	NiFi integrates smoothly with existing IT infrastructure, supporting seamless data exchange.	Should-have
COM-06	The tool supports connectivity to various systems and databases without any compatibility issues.	Should-have
SEC-04	All data transfers through NiFi are encrypted, ensuring secure communication.	Must-have
SEC-07	Periodic saving of data is ensured to prevent data loss in case of a cyber-attack.	Must-have
SEC-11	NiFi has robust backup functions to ensure data can be restored when needed.	Must-have

7.3.4 Sequence Diagram

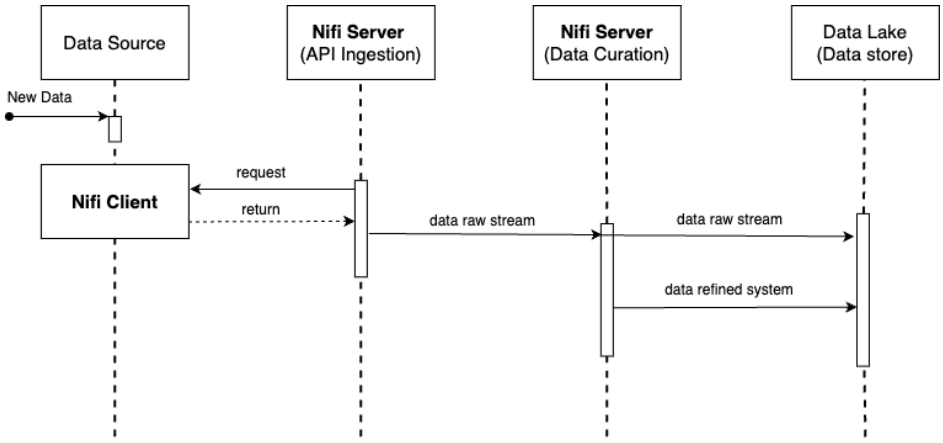

Figure 26. Apache Nifi Sequence Diagram

7.4 API Gateway

7.4.1 Description

The API Gateway is a crucial component of the FALCON platform, serving as the entry point for all client requests to the backend services. It acts as a reverse proxy, routing requests, enforcing security policies, managing traffic, and providing other cross-cutting concerns such as authentication and rate limiting. By centralizing API management, the API Gateway simplifies the architecture and ensures consistent policy enforcement across all services.

Within the FALCON framework, the API Gateway facilitates seamless integration between the several tools and external systems. It ensures that all communications are secure, and that data integrity is maintained throughout the transaction processes. The API Gateway also provides detailed analytics and monitoring capabilities, allowing the FALCON team to track API usage and performance, identify issues, and optimize the system's functionality.

7.4.2 Functional Requirements

Table 18. API Gateway – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	The API Gateway successfully connects to all existing databases and services as per configuration.	Must-have
FUN-05	Ensures secure and efficient routing of multimedia content (photos, video, audio) requests.	Must-have
FUN-11	The system functions independently from the current infrastructure, with the API Gateway managing traffic.	Must-have

7.4.3 Non-Functional Requirements

Table 19. API Gateway – Non-functional Requirements

ID	Acceptance criteria	Ranking
SEC-01	Implements role-based access control (RBAC) to restrict access based on user roles and responsibilities.	Must-have
SEC-02	Maintains a detailed data access log for traceability purposes.	Must-have
SEC-03	Enforces security codes and access control regimes, managing allocation of roles.	Must-have
SEC-04	Encrypts data in transit between the client and server, ensuring secure communication.	Must-have
SEC-06	Supports different levels of access (e.g., administrator and user roles) effectively.	Must-have
SEC-13	Implements two-factor authentication (2FA) for user login to enhance security.	Should-have
SEC-14	Tracks internal usage statistics and provides documentation to prevent abuse of the system.	Should-have
COM-01	Compatible with multiple communication technologies (fibre, 4G/5G, TETRA, satellite).	Must-have
COM-04	Provides real-time notifications for critical updates or changes.	Should-have
COM-06	Ensures connectivity and interoperability with different systems and databases.	Should-have

7.4.4 Sequence Diagram

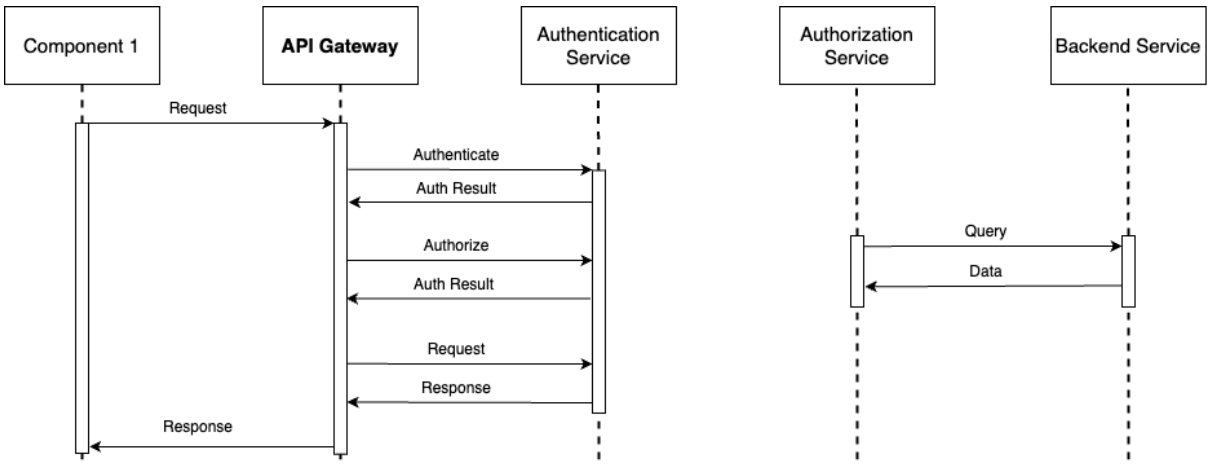

Figure 27. API Gateway Sequence Diagram

7.5 CI/CD Platform

7.5.1 Description

GitLab CI/CD is a robust CI/CD tool integrated into the GitLab platform. It automates the process of software integration, testing, and deployment, ensuring that code changes are systematically verified and deployed in a reliable and repeatable manner. Within the FALCON platform, GitLab CI/CD facilitates the development lifecycle by enabling frequent, reliable releases and ensuring that all code changes pass through rigorous testing before being deployed.

GitLab CI/CD supports the FALCON platform by automating the build, test, and deployment processes. This ensures that any changes made to the codebase are automatically tested and deployed, reducing the risk of human error, and enhancing the overall efficiency of the development process. The use of GitLab CI/CD within FALCON also provides detailed logging and monitoring capabilities, helping the development team quickly identify and resolve any issues that arise during the CI/CD pipeline.

7.5.2 Functional Requirements

Table 20. CI/CD Platform – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-01	The user manual includes detailed instructions for using GitLab CI/CD pipelines.	Must-have
FUN-02	The interface for managing CI/CD pipelines is user-friendly and accessible to all team members.	Must-have
FUN-11	The CI/CD system functions independently, ensuring consistent operation regardless of infrastructure changes.	Must-have

7.5.3 Non-Functional Requirements

Table 21. CI/CD Platform – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	GitLab CI/CD implements a robust backup and recovery system to ensure continuity in case of failures.	Must-have
OPE-02	The platform includes a user-friendly interface with customizable dashboards for monitoring CI/CD pipelines.	Must-have
OPE-08	Supports real-time data updates and provides traceability of CI/CD steps for auditing purposes.	Should-have
SEC-04	All data transfers during the CI/CD processes are encrypted, ensuring secure communication.	Must-have
SEC-05	Regular security audits of the CI/CD pipelines are conducted to identify and mitigate vulnerabilities.	Must-have

SEC-07	Periodic saving of CI/CD logs and results to prevent data loss in case of a cyber-attack.	Must-have
COM-06	Ensures connectivity and interoperability with different systems and databases for deployment.	Should-have
COM-07	Implements effective warnings and alarm systems for immediate attention to CI/CD pipeline issues.	Should-have

7.5.4 Sequence Diagram

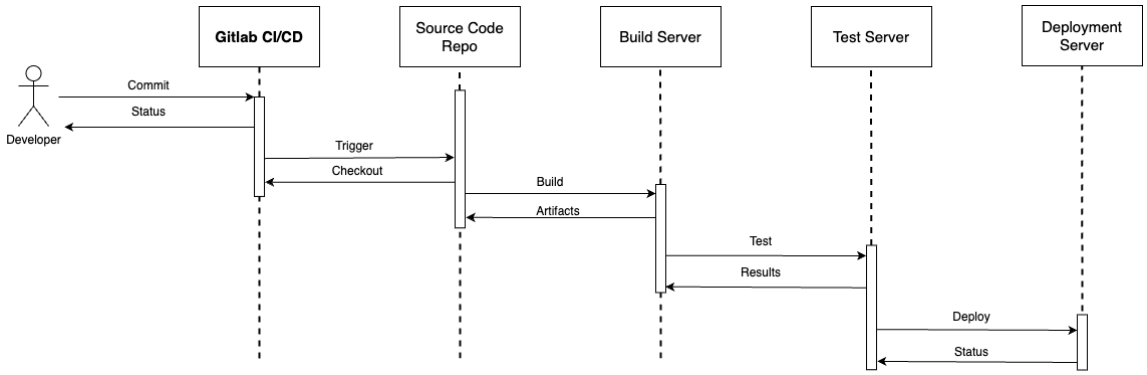

Figure 28. CI/CD Platform Sequence Diagram

7.6 Keycloak

7.6.1 Description

Keycloak is an open-source identity and access management tool that provides robust authentication and authorization services. Within the FALCON platform, Keycloak is essential for managing user identities, securing access to applications, and enforcing security policies across the system. It supports single sign-on (SSO), user federation, client adapters, and social login, among other features.

By integrating Keycloak, FALCON ensures that only authorized users can access sensitive data and services, enhancing the overall security of the platform. Keycloak's ability to manage user roles and permissions dynamically allows for granular access control, ensuring that users can only access the resources they are permitted to. Additionally, Keycloak provides detailed logging and monitoring capabilities, which are crucial for auditing and compliance purposes.

7.6.2 Functional Requirements

Table 22. Keycloak – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	Keycloak successfully connects to all existing databases as per the configuration and access permissions.	Must-have

FUN-07	Data from external sources is consistently and accurately ingested into the platform.	Must-have
FUN-08	Keycloak can handle and process large volumes of "Big Data" without performance degradation.	Must-have
FUN-11	The system continues to operate independently from current infrastructure, even during high data loads.	Must-have

7.6.3 Non-Functional Requirements

Table 23. Keycloak – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	Keycloak ensures continuity in case of failures with a robust backup and recovery system.	Must-have
OPE-08	Supports real-time data updates and provides traceability of access and authorization steps for auditing purposes.	Should-have
SEC-01	Role-based access control (RBAC) is implemented, restricting access based on user roles and responsibilities.	Must-have
SEC-02	Detailed data access logs are maintained for traceability and auditing.	Must-have
SEC-03	Security codes and access control regimes are enforced, managing the allocation of roles.	Must-have
SEC-04	Data in transit between the client and server is encrypted, ensuring secure communication.	Must-have
SEC-05	Regular security audits are conducted to identify and mitigate vulnerabilities.	Must-have
SEC-06	Supports different levels of access (e.g., administrator and user roles) effectively.	Must-have
SEC-08	User data is encrypted when stored in the database.	Should-have
SEC-10	Implements effective warnings and alarm systems for immediate attention to critical security events.	Should-have
SEC-13	Two-factor authentication (2FA) is implemented for user login to enhance security.	Should-have

7.6.4 Sequence Diagram

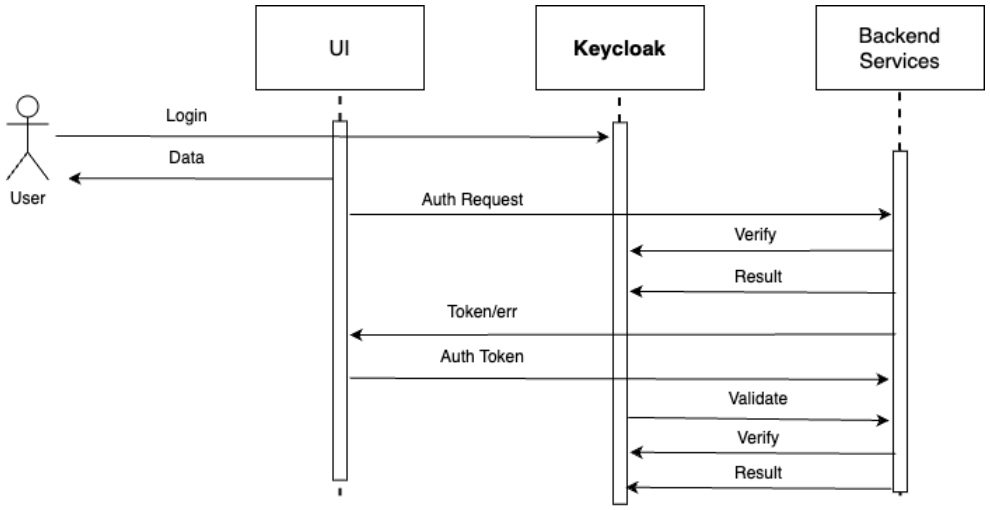

Figure 29. Keycloak Sequence Diagram

7.7 OpenVPN

7.7.1 Description

OpenVPN is a robust and flexible open-source VPN solution that provides secure communication across the internet by creating encrypted tunnels between devices. Within the FALCON platform, OpenVPN is utilized to ensure secure, encrypted connections between remote users and the FALCON system, safeguarding data integrity and privacy during transmission. This tool is essential for enabling secure remote access, protecting sensitive information from potential interception or unauthorized access.

OpenVPN supports the FALCON platform by facilitating secure remote connections for users, ensuring that all data transmitted between the user's device and the FALCON infrastructure is encrypted and secure. This capability is crucial for maintaining the confidentiality and integrity of sensitive data, especially when users are accessing the system from remote or unsecured locations. OpenVPN's strong encryption and authentication mechanisms help protect against various cyber threats, ensuring that only authorized users can access the platform.

7.7.2 Non-Functional Requirements

Table 24. OpenVPN – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	OpenVPN ensures continuity in case of failures with a robust backup and recovery system.	Must-have
SEC-04	All data transfers through OpenVPN are encrypted, ensuring secure communication.	Must-have
SEC-05	Regular security audits are conducted to identify and mitigate vulnerabilities in the VPN setup.	Must-have

SEC-07	Periodic saving of connection logs and integrity checks confirm no data loss during cyber-attack simulations.	Must-have
SEC-11	Regular backups of VPN configurations and logs are completed without errors.	Must-have
SEC-12	Encrypted storage of VPN logs to protect sensitive connection data.	Should-have
SEC-13	Two-factor authentication (2FA) is implemented for VPN login to enhance security.	Should-have
COM-01	Compatible with multiple communication technologies (fibre, 4G/5G, TETRA, satellite)	Must-have
COM-06	Ensures connectivity and interoperability with different systems for secure access	Should-have

7.7.3 Sequence Diagram

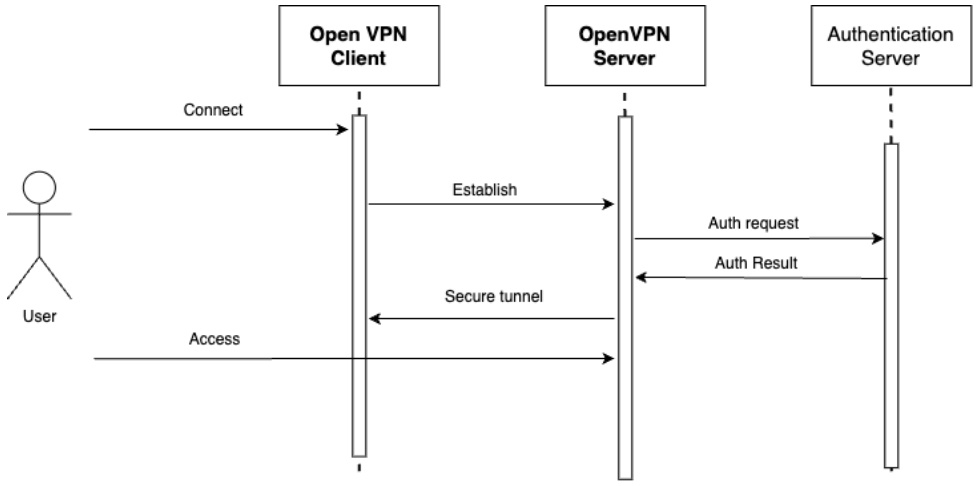

Figure 30. OpenVPN Sequence Diagram

7.8 RKE2

7.8.1 Description

RKE2 (Rancher Kubernetes Engine 2) is a lightweight, highly available Kubernetes distribution designed for deploying and managing containerized applications. Within the FALCON platform, RKE2 is utilized to orchestrate the deployment, scaling, and management of containerized services and applications. It provides a robust infrastructure for automating deployment processes, ensuring that applications are consistently and reliably delivered across various environments.

RKE2 supports the FALCON platform by enabling seamless deployment of containerized applications and services. It ensures that all components of the platform are deployed in a consistent and efficient manner, facilitating scalability and high availability. With

RKE2, the FALCON platform can leverage Kubernetes' powerful orchestration capabilities to manage resources effectively, automate application updates, and ensure continuous delivery of new features and fixes. Additionally, RKE2 enhances the platform's resilience by providing built-in mechanisms for load balancing, service discovery, and fault tolerance.

7.8.2 Functional Requirements

Table 25. RKE2– Functional Requirements

ID	Acceptance criteria	Ranking
FUN-11	The system continues to operate independently from current infrastructure, with RKE2 managing container orchestration.	Must-have

7.8.3 Non-Functional Requirements

Table 26. RKE2 – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	RKE2 ensures continuity in case of failures with a robust backup and recovery system.	Must-have
OPE-06	Supports automated scaling of containerized applications based on workload demands.	Should-have
OPE-08	Provides real-time data updates and monitoring of containerized applications for auditing purposes.	Should-have
COM-03	RKE2 integrates smoothly with existing IT infrastructure, supporting seamless deployment processes.	Should-have
COM-06	The tool supports connectivity to various systems and databases without any compatibility issues.	Should-have
COM-07	Implements effective warnings and alarm systems for immediate attention to deployment issues.	Should-have
SEC-07	Data is periodically saved, and integrity checks confirm no data loss occurs during cyber-attack simulations.	Must-have
SEC-11	Regular backups of container configurations and logs are completed without errors.	Must-have

7.8.4 Sequence Diagram

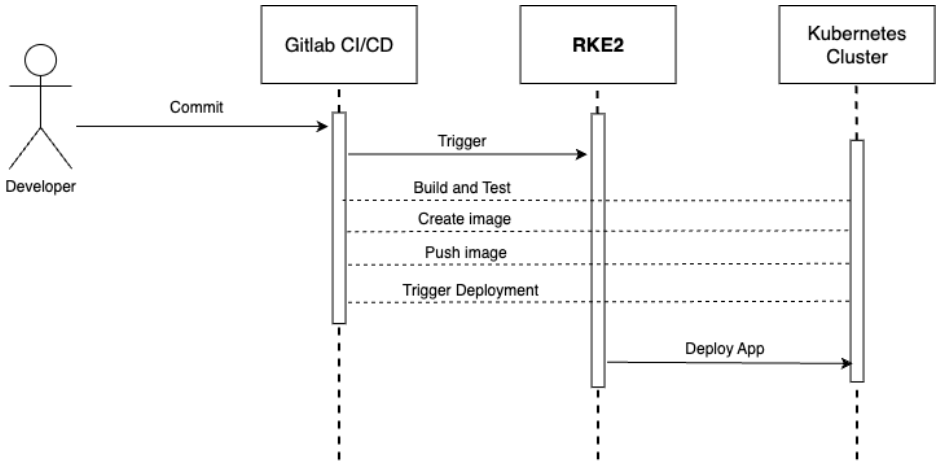

Figure 31. RKE2 Sequence Diagram

7.9 Trend Detection

7.9.1 Description

The FALCON Trend Detection (FTD) tool is, designed to focus on the identification of possible trends in multimodal data coming from heterogeneous sources (e.g., cryptocurrency markets) related to the existence of potential corruption phenomena (e.g., money laundering). In particular, spatial and temporal information will be exploited for more reliable detection of trends, patterns and possible anomalies that may suggest links to corruption activities. By leveraging comprehensive data integration, and machine learning techniques (e.g., time series analysis methods), FTD will deliver in-depth trend analysis focusing on historical and real-time (if possible) trend insights. The details of the development of the FTD tool will be reported in D4.2-D4.4.

7.9.2 Functional Requirements

Table 27. FALCON Trend Detection – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-13	System should include ML capabilities that can be trained and improved over time.	Should-have

7.9.3 Non-Functional Requirements

Table 28. FALCON Trend Detection – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-07	Support of real-time data streams.	Should-have

7.9.4 Sequence Diagram

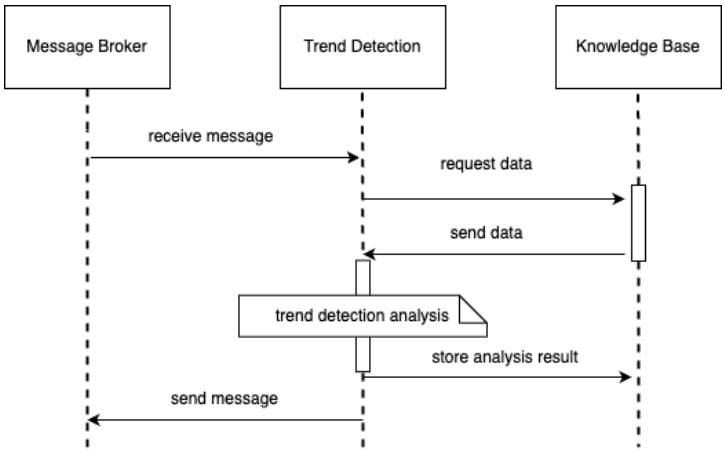

Figure 32. Trend Detection Sequence Diagram

7.10 Predictive Analytics

7.10.1 Description

The FALCON Predictive Analytics (FPA) tool is an advanced analytical tool designed to forecast corruption risk, fraudulent behaviour and cryptocurrency-related insights in public procurement processes. At its core, FPA is dedicated to proactively identifying and neutralizing risks in both traditional procurement and cryptocurrency transactions. Leveraging cutting-edge machine learning algorithms, comprehensive data integration, and advanced analytics, FPA provides actionable insights to enhance transparency, integrity, and accountability in procurement operations, while also addressing emerging challenges and opportunities in the cryptocurrency space. The details of the development of the FPA tool will be reported in D5.1-D5.3.

7.10.2 Functional Requirements

Table 29. FALCON Predictive Analytics– Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have

FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-13	System should include ML capabilities that can be trained and improved over time.	Should-have

7.10.3 Non-Functional Requirements

Table 30. FALCON Predictive Analytics – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-07	Support of real-time data streams.	Should-have

7.10.4 Sequence Diagram

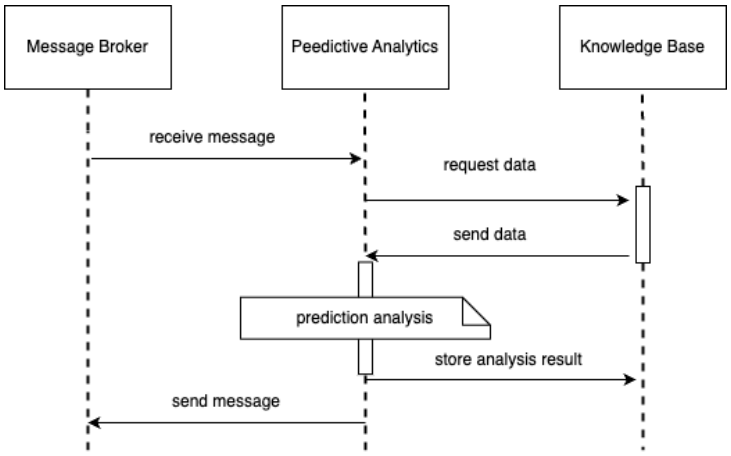

Figure 33. Predictive Analysis Sequence Diagram

7.11 Border Corruption Investigation

7.11.1 Description

The FALCON Border Corruption Investigation (FBCI) tool is, designed to focus on the identification of possible trends in multimodal data coming from heterogeneous sources (Tabular and Video Stream) related to the existence of potential corruption phenomena (e.g., smuggling). Spatial and temporal information will be exploited for more reliable detection of trends, patterns and possible anomalies that may suggest links to organized crime at the border control points. By leveraging comprehensive data integration, and machine learning techniques (e.g., time series analysis methods), FBCI will deliver in-depth trend analysis focusing on historical and real-time (if possible) trend insights. The details of the development of the FBCI tool will be reported in D4.2-D4.4.

7.11.2 Functional Requirements

Table 31. FALCON Border Corruption Investigation tool – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-09	Support concurrent access by multiple users without degradation in system performance.	Must-have
FUN-10	The system must provide multi-language support to accommodate users in different regions and ensure inclusivity and accessibility.	Must-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have
FUN-13	System should include ML capabilities that can be trained and improved over time.	Should-have

7.11.3 Non-Functional Requirements

Table 32. FALCON Border Corruption Investigation tool – Non-functional Requirements

ID	Acceptance criteria	Ranking
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have
SEC-03	Enhanced security measures must be in place to protect sensitive data from unauthorized access.	Must-have
SEC-10	System must be robust and maintain operational capacity under various stress conditions.	Must-have
OPE-02	Interface must allow customization to meet the specific needs of different user types.	Could-have
OPE-04	Search functionality must be robust and capable of handling complex queries efficiently.	Must-have
COM-03	System must ensure seamless integration with different IT environments and support various data formats.	Should-have

COM-07	System should have mechanisms to alert users about system states, errors, or important notifications.	Should-have

7.11.4 Sequence Diagram

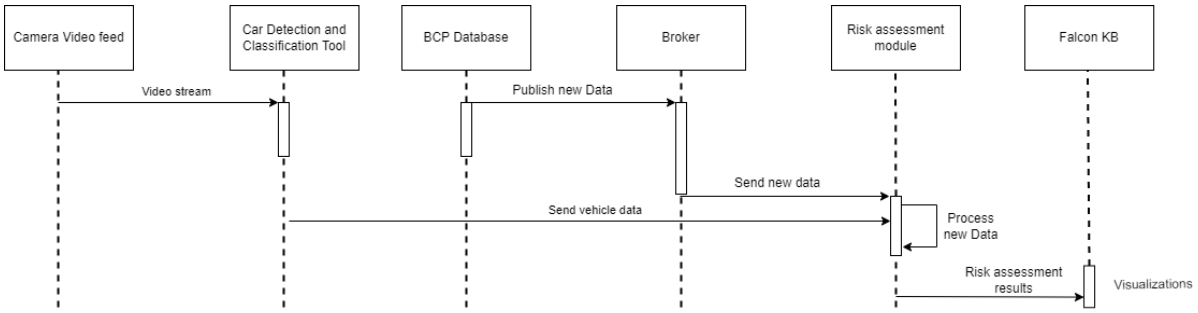

Figure 34. Border Corruption Investigation Tool Sequence Diagram

7.12 Investigative Tool for Corruption Cases

7.12.1 Description

The proposed tool is designed to assist LEAs in combating corruption and economic crimes through advanced technological integration and user-friendly interfaces. This tool encompasses both backend and frontend components that seamlessly work together to enhance the efficiency and effectiveness of criminal investigations.

Backend capabilities:

The tool's backend is strong and designed to communicate extensively with the FALCON KB, which serves as the main hub for gathered data. The two primary functions that this backend architecture is intended to support are:

1. **Search and Retrieval Connection to FALCON KB:** The backend is set up to use API calls to establish a direct connection to the FALCON knowledge base. The retrieval of data housed within the Knowledge Base (KB), depends on this link. The set of data stored in the KB are navigated through by the backend using queries, which facilitates rapid and accurate data extraction. This feature makes sure that the front-end display complete and current data.
2. **Pattern Identification in Graphs:** The obtained data will be processed using various approaches to find hidden patterns and abnormalities within the intricate networks that are depicted in the graphs. The algorithms utilize the indices created in earlier development stages and are based on the shared representational model. Suspicious patterns can be found by the backend by examining the connections and data flow between nodes. Furthermore, the tool will provide an ongoing means of updating itself with fresh inputs and operator feedback, guaranteeing its capacity to adjust to changing criminal tactics.

Frontend capabilities:

The frontend of the tool is designed to be intuitive and interactive, focusing on the visualization aspects to aid LEAs in data interpretation and decision-making processes:

1. **User-Friendly Dashboards:** The dashboards, which are the main component of the frontend, are made to provide several viewpoints and aggregate views of the data that the backend has gathered and examined. These dashboards provide broad overviews of criminal networks and operations as well as granular views of aspects, allowing users to dive deep into the data.
2. **Visualization functionalities:** To improve the investigative process, the frontend has interactive functionalities that let investigators work with and examine data in different forms in addition to displaying it. Users can customize the data display based on specific requirements or research objectives by utilizing features like filter, sort, and zoom.
3. **Customized Views of Big Graphs:** the tool offers services for large-scale graph views that may be customized to represent complicated datasets. These views facilitate the visualization of relationships and patterns in the data, which helps analysts identify anomalies and follow criminal linkages across large databases.

More details about this tool, including the final name of the tool, will be described in D5.2 and D5.3.

7.12.2 Functional Requirements

Table 33. Investigative tool for corruption cases – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-01	Easy to use user manual	Must-have
FUN-02	User-friendly interface	Must-have
FUN-10	Local language support (e.g. Romanian/ English/ French/ German)	Must-have
FUN-14	Modest system requirements for end-user device	Should-have

7.12.3 Non-Functional Requirements

Table 34. Investigative tool for corruption cases – Non-functional Requirements

ID	Acceptance criteria	Ranking
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have

SEC-07	Periodic saving of data in order to prevent data lost in case of a cyber-attack	Must-have

7.12.4 Sequence Diagram

The diagram depicted below outlines the sequence diagram for the proposed tool. On the backend side, the application handles interactions with the FALCON Knowledge Base and oversees retrieving information. It is also equipped to apply specific search criteria through the KB APIs to extract pieces of information. After the information has been retrieved, it undergoes processing—for instance, identifying patterns—and is subsequently displayed on the user interface for visualization. This sequence ensures that data not only is efficiently gathered and refined but also is presented in an accessible manner to users through the front-end of the application.

As of the date of this document, WP5 has not yet commenced. Therefore, modifications or upgrades to the current tool may be implemented as the project progresses. More detailed information about these changes will be provided in the forthcoming deliverables of WP5: D5.2 and D5.3.

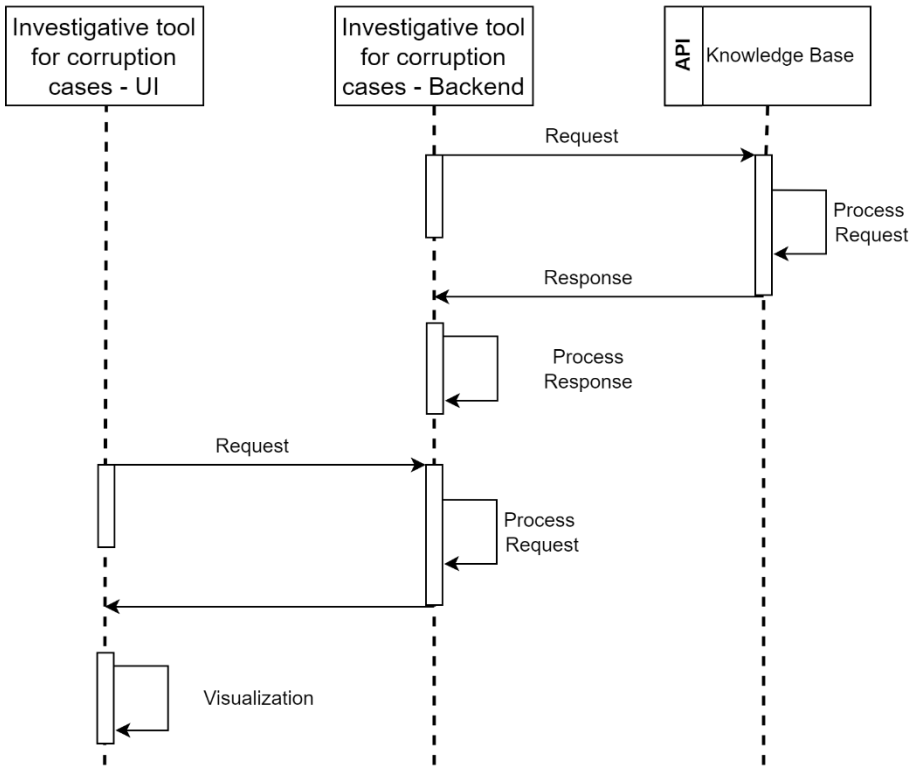

Figure 35. Investigative Tool for Corruption Cases – Sequence Diagram

7.13 Car Detection and Classification Tool

7.13.1 Description

The Car Detection and Classification Tool is designed to assist LEAs in fighting corruption offenses via utilizing state-of-the-art Artificial Intelligence techniques. More specifically, it is used in the context of Use Case 3 – Tracking and Investigating Corruption Schemes in Border Control Points. The tool analyses a video stream with the objective of detecting each car crossing the Border Control Points (BCPs) and identifying its model, make, year of manufacture and colour. A tracker is incorporated in the method as well in order to associate each car with a unique identity. Subsequently, all the extracted elements are stored to the Knowledge Base thereby furnishing a comprehensive compilation of information pertaining to each car.

7.13.2 Functional Requirements

Table 35. Car Detection and Classification Tool – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-05	Method should analyse each video frame and merge the results of the image sequence to improve the resulted output.	Must-have
FUN-11	The system should run independently without requiring any additional libraries/packages etc.	Must-have
FUN-13	The module should be based on DL algorithm able to be retrained on different/extended datasets to improve accuracy.	Should-have
FUN-14	The module will be able to run on GPU/ non-GPU setups allowing it to be operated in end-user devices.	Should-have
FUN-17	The module should be able to analyse videos from recorded video files or/and camera streams.	Could-have

7.13.3 Non-Functional Requirements

Table 36. Car Detection and Classification Tool – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-07	The system will be able to process real-data stream and provide results during and after the end of each car visit.	Should-have

7.14 License Plate Detection and Recognition Tool

7.14.1 Description

The License Plate Detection and Recognition Tool follows the Car Detection and Classification Tool with the objective of detecting and recognizing the license plates of each car crossing the Border Control Points (BCPs). This information – license plate

number, is also stored in the Knowledge Base, thereby enhancing the overall data associated with each car.

7.14.2 Functional Requirements

Table 37. License Plate Detection and Recognition Tool – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-05	Method should analyse each video frame and merge the results of the image sequence to improve the resulted output.	Must-have
FUN-11	The system should run independently without requiring any additional libraries/packages etc.	Must-have
FUN-13	The module should be based on DL algorithm able to be retrained on different/extended datasets to improve accuracy.	Should-have
FUN-14	The module will be able to run on GPU/ non-GPU setups allowing it to be operated in end-user devices.	Should-have
FUN-17	The module should be able to analyse videos from recorded video files or/and camera streams.	Could-have

7.14.3 Non-Functional Requirements

Table 38. License Plate Detection and Recognition Tool – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-07	The system will be able to process real-data stream and provide results during and after the end of each car visit.	Should-have

7.14.4 Sequence Diagram

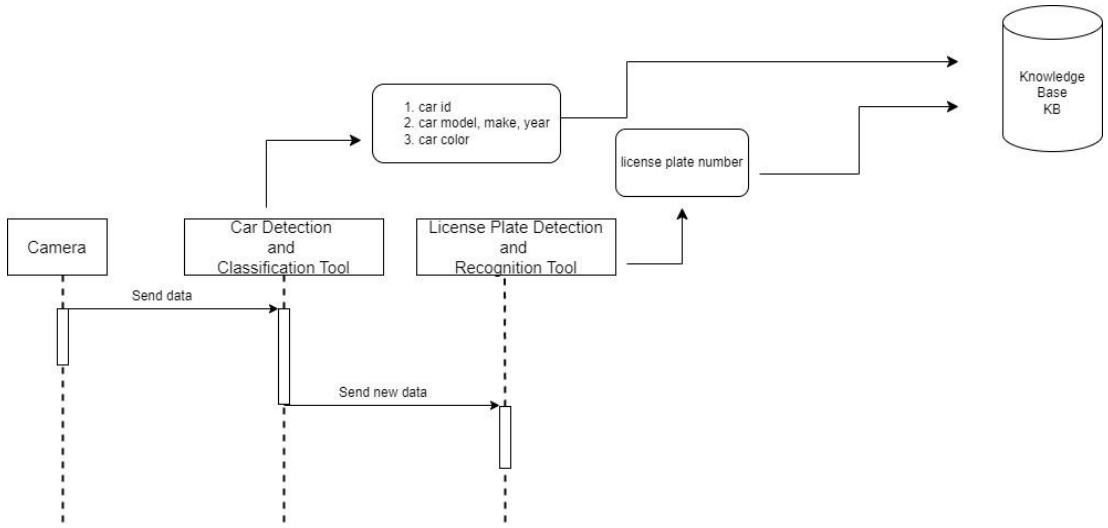

Figure 36. Car Detection and Classification Tool combined with License Plate Detection and Recognition Tool Sequence Diagram

7.15 Advanced Corruption Risk Assessment Tool

7.15.1 Description

The Advanced Corruption Risk Assessment Tool will be designed to provide a comprehensive evaluation of corruption risks by assessing the likelihood and potential impact of various risk factors. This tool will leverage outputs from WP2, including measures and indices of corruption and impact assessment methods, combined with identified data sources and corruption indicators. By integrating these diverse data sources, the tool will enable users to model and simulate risks through a sophisticated user interface. At its core, the tool will feature a risk modelling component supported by a model graph and simulation capabilities, facilitating the assessment and prediction of corruption risks. Users will be able to select from various risk scenarios and estimate the likelihood and impact of corruption based on different parameters, empowering them to make informed decisions based on reliable risk assessments.

7.15.2 Functional Requirements

Table 39. Advanced Corruption Risk Assessment – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-01	The tool must provide an easy-to-use user manual.	Must-have
FUN-02	The tool must provide a user-friendly interface for risk modelling that allows users to input data and select risk scenarios.	Must-have
FUN-03	System must integrate seamlessly with existing databases using secure and efficient protocols, to extract relevant corruption indicators.	Must-have
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-05	The tool must support the input of various data formats and sources, ensuring compatibility and accurate integration of external data.	Should-have
FUN-06	The tool must calculate the total risk of corruption by assessing the likelihood and associated impact of each identified risk.	Must-have
FUN-07	The tool must me capable of gathering data from external sources, supporting the input of various data formats and sources.	Must-have
FUN-09	Support concurrent access by multiple users without degradation in system performance.	Must-have
FUN-10	Advanced Corruption Risk Assessment tool must provide multi-language support to accommodate users in different regions.	Could-have
FUN-11	System must operate in a standalone mode with all necessary functionalities without external dependencies.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have

FUN-13	The tool must support multiple ML approaches for risk modelling including logistic regression, Markov models, and neural networks	Could-have

7.15.3 Non-Functional Requirements

Table 40. Advanced Corruption Risk Assessment – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	Tool must enforce user access based on predefined roles, ensuring secure access control.	Must-have
OPE-02	Tool's user interface must allow customization to meet the specific needs of different user types. Moreover, assessment events must be logged.	Must-have
OPE-03	The tool's interface must support the visualization of model graphs and simulation components to help users understand the risk assessment process.	Must-have
OPE-04	The tool must provide advanced searching and filtering capabilities to search information during operation (keyword search / refined search).	Must-have
OPE-05	Risk assessment procedures provided for company data and procurement data.	Must-have
OPE-07	Tool must support real time data streams with minimal latency in data processing and report generation. It should be capable of handling large datasets efficiently without significant degradation in performance.	Should-have
OPE-09	Tool must be designed to scale seamlessly as the amount of data and number of users grows. The tool must be scalable to accommodate increasing amounts of data and additional computational complexity as more models and scenarios are added	Should-have
OPE-15	Tool could be installable in different devices (PC/ Tablet / Smartphone).	Could-have
COM-01	Tool must support specified communication technologies to ensure wide accessibility and connectivity.	Must-have
COM-02	Tool can ensure communication between all persons or LEAs involved in a specific case.	Should-have
COM-03	Tool must ensure seamless integration with different IT environments and support various data formats.	Should-have
COM-06	Tool should provide support of connectivity to different systems and databases.	Should-have
SEC-01	Tool must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	Tool must log all data access events to enable audit and traceability.	Must-have

SEC-03	Enhanced security measures must be in place to protect sensitive data and assessment results from unauthorized access.	Must-have

7.15.4 Sequence Diagram

The Advanced Corruption Risk Assessment Tool will consist of three main modules: an API, a UI, and a KB. For each of the functionalities, the process follows the schema illustrated in the following figure. The user interacts with the tool through the User Interface, which communicates with the API. The API is responsible for importing, computing, and exporting the results, making them available for use by other tools or APIs. Additionally, the API connects with the Knowledge Database to retrieve and utilize the corruption indicators that have been populated within it. This design ensures that end-users can easily interact with the tool through the UI, while the API handles data processing and integration with external systems.

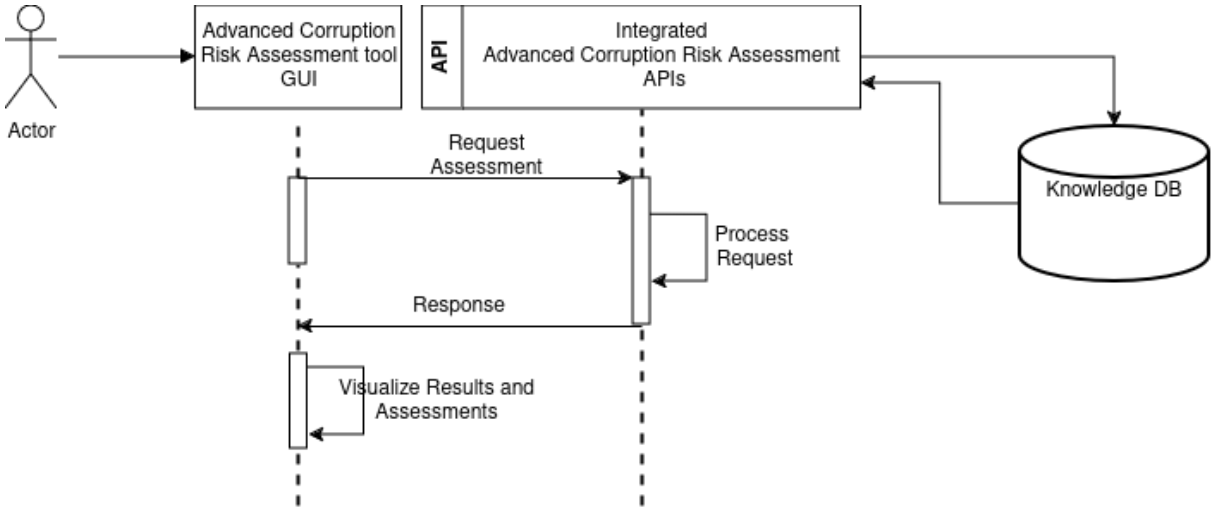

Figure 37. Advanced Corruption Risk Assessment Sequence Diagram

7.16 OSINT Tool

7.16.1 Description

The OSINT Tool will be used as a data acquisition service of publicly available online content, pertinent to financial corruption. The data will be collected based on specified risk terms (keywords and key phrases) and will be used for creating datasets available for search and analysis, helping LEAs in crime investigation and threat & risk assessment, as they will be consumed by the analytics tools of FALCON for the extraction of valuable intelligence. This service will accommodate use cases 1, 3 and 4. The online sources that will be considered will be Clearnet and Facebook. Regarding the first source, specific starting points (seeds) for the web crawlers will be defined by the end users, like publicly available web sites containing information that could be used to trace public corruption (use case 1) and conflicts of interest of politically exposed persons (use case 4). Considering Facebook, synthetic profiles will be created and populated with content

related to use case 3, to demonstrate the usage of social media data for assessing border guards' social status in terms of inconsistent lifestyle. This approach will be followed to address privacy/ethics requirements associated with the consent of the data subjects to process their social media profiles.

The OSINT Tool will be modular, comprising of:

- a Data Module, accepting OSINT data requests and transferring OSINT data to the FALCON Knowledge Base for risk assessment and further analysis,
- a Gateway, communicating with external web crawlers for OSINT data collection,
- an OpenSearch repository for indexing the crawled data into datasets,
- a UI for setting the targets to be crawled and viewing crawled data.

7.16.2 Functional Requirements

Table 41. OSINT – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-04	Functions must provide accurate, reliable results, and handle multiple queries simultaneously.	Must-have
FUN-08	System must efficiently process and analyse large datasets within specified performance benchmarks.	Must-have
FUN-09	Support concurrent access by multiple users without degradation in system performance.	Must-have
FUN-10	System must provide multi-language support to accommodate users in different regions.	Must-have
FUN-12	Database must support high transaction rates and complex queries with minimal latency.	Must-have

7.16.3 Non-Functional Requirements

Table 42. OSINT – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	System must have failover capabilities and rapid recovery methods to ensure data integrity and availability.	Must-have
OPE-02	Interface must allow customization to meet the specific needs of different user types.	Must-have
OPE-09	System must be designed to scale seamlessly as the amount of data and number of users grows.	Should-have
COM-01	System must support specified communication technologies to ensure wide accessibility and connectivity.	Must-have
COM-03	System must ensure seamless integration with different IT environments and support various data formats.	Should-have

COM-06	The platform must be able to connect with various external systems and databases efficiently.	Should-have
SEC-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
SEC-02	System must log all data access events to enable audit and traceability.	Must-have
SEC-03	Enhanced security measures must be in place to protect sensitive data from unauthorized access.	Must-have
SEC-10	System must be robust and maintain operational capacity under various stress conditions.	Must-have

7.16.4 Sequence Diagram

The OSINT tool will accept data requests either through a Message Broker or directly through a REST API, collect the requested data from the web crawlers, process and index them locally and finally store them in the Knowledge Base for further analysis.

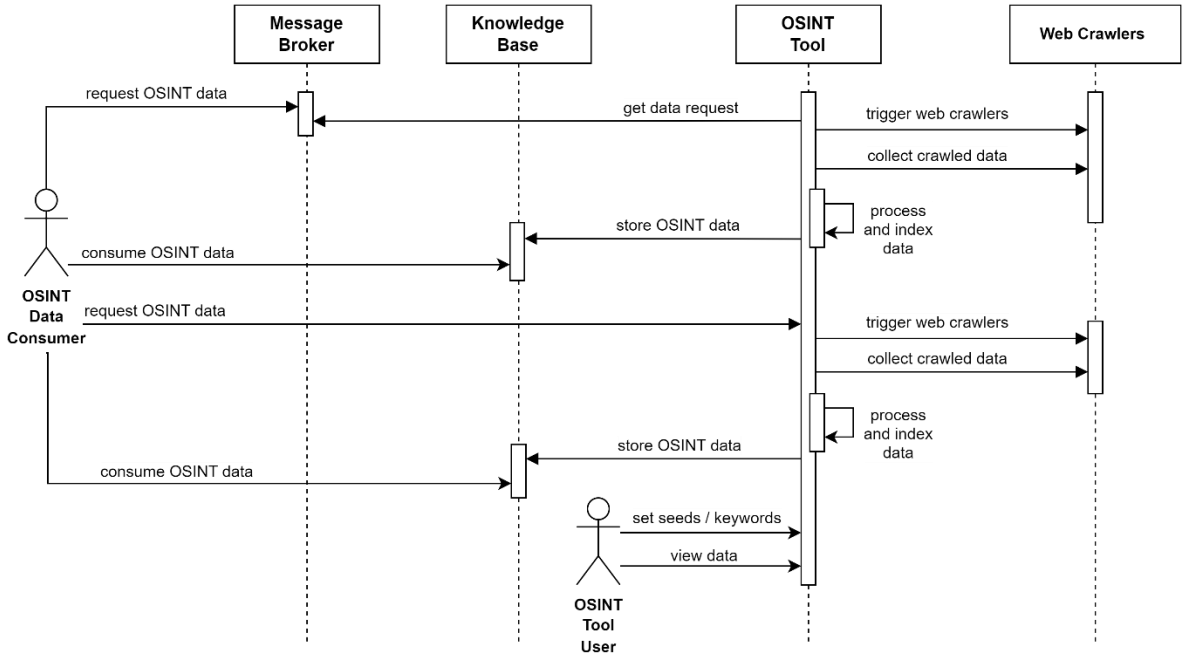

Figure 38. OSINT Sequence Diagram

7.17 Kriptosare

7.17.1 Description

Kriptosare makes use of state-of-the-art Artificial Intelligence techniques to analyse the blockchain and learn how to identify and highlight the most important red flag indicators that could suggest criminal behaviour, such as anonymity-enhanced cryptocurrencies and tumbling services. Kriptosare is a tool that allows us to analyse the different behaviours adopted by users of various cryptocurrencies, mainly Bitcoin and Litecoin. More specifically, Kriptosare is based on a Vicomtech library/tool capable of combining

OSINT information and the power of Machine Learning models for classifying the interaction and dynamics (i.e. behaviours) that different entities and addresses generate in the blockchain. Furthermore, using external information, Kriptosare aims to highlight if an address belongs to a punished entity (e.g., under OFAC sanction) and if it has been used after the sanction.

7.17.2 Functional Requirements

Table 43. Kriptosare – Functional Requirements

ID	Acceptance criteria	Ranking
FUN-01	System must provide transaction statistics regarding the information of a searched crypto addresses	Must-have
FUN-02	System must provide entity statistics regarding the information of a searched crypto addresses	Must-have
FUN-03	System must provide a classification of an addresses over-time, at least for Bitcoin and Litecoin network	Must-have
FUN-04	System must provide a classification of an addresses in a specific temporal point, at least for Bitcoin and Litecoin network	Must-have
FUN-05	System must provide a list of the available behaviour classified in the blockchain	Must-have
FUN-06	System must provide a similarity score related to the classification of a sample in a specific behaviour	Must-have
FUN-06	The tool should show a grade of a searched address related to the risk of being involved in illicit/fraud/scam transactions	Should-have

7.17.3 Non-Functional Requirements

Table 44. Kriptosare – Non-functional Requirements

ID	Acceptance criteria	Ranking
OPE-01	System must enforce user access based on predefined roles, ensuring secure access control.	Must-have
OPE-02	Interface must allow customization to meet the specific needs of different user types. System must log all data access events to enable audit and traceability.	Must-have
OPE-03	The tool shall allow users to export data using open data exchange formats. At minimum JSON, and if possible, also XML and/or CSV	Must-have
OPE-04	The tool shall provide a self-explanatory user interface	Must-have
OPE-05	The tool shall present to the user understandable messages to guide and facilitate the operation of the tool.	Must-have

OPE-06	In the event of an interruption or a failure, the tool should recover the data affected and re-establish the state of the system.	Must-have
OPE-07	System must ensure seamless integration with different IT environments and support various data formats.	Should-have
OPE-08	The tool should not degrade its performance (e.g., errors, response time) when the number of parallel request or sessions	Should-have
OPE-09	Tool must be designed to scale seamlessly as the amount of data and number of users grows.	Should-have

7.17.4 Sequence Diagram

Kriptosare is composed of three main modules: interface (kripto_viz), API and the Database (DB). For this reason, for each of the functionalities it follows the schema (diagram) reported in the following Figure. Indeed, the user makes a request through the interface, which in turn calls the API, that consults the information in the DB and return the information to be visualized and processed by the final user.

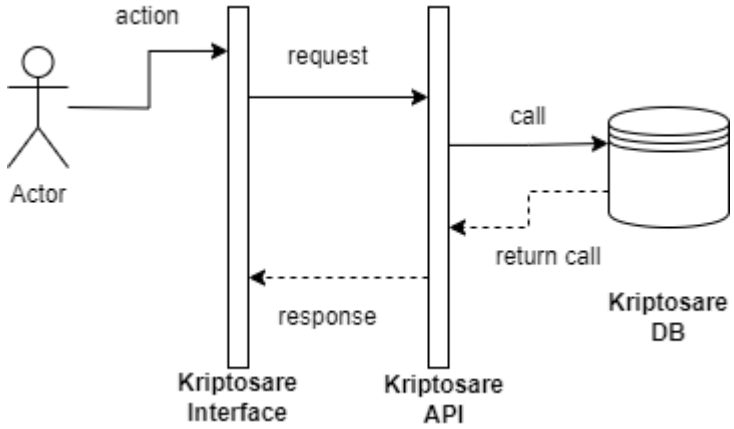

Figure 39. Kriptosare Sequence Diagram

8. Summary and Conclusions

D3.2 "FALCON Framework Architecture", has provided a comprehensive overview of the architectural design and methodological approaches adopted for the development of the FALCON system. The detailed specifications, validated by stakeholders, outline both the functional and non-functional requirements, ensuring that the system aligns with the project's overarching goals and user needs. By leveraging an agile methodology and integrating a CI/CD pipeline using GitLab CI/CD, we have established a robust framework that facilitates continuous improvement, seamless updates, and thorough testing of all components.

The high-level architecture presented in this deliverable covers core components, external interfaces, and platform requirements, illustrating how they interact and exchange data. Communication, authorization, and deployment strategies have been meticulously detailed, incorporating modern infrastructure technologies like Kubernetes to ensure scalability, security, and resilience. Additionally, the secure management of data and the integration of trustworthy AI principles have been emphasized to maintain the integrity and effectiveness of the FALCON platform.

Each tool within the FALCON toolkit has been described in detail, highlighting its roles, functionalities, and how it contributes to the overall system. This comprehensive description ensures that all technological components are well-understood and effectively integrated to achieve the project's objectives.

As this is the first iteration of the framework architecture, it represents a foundational step in the FALCON project's development. As the project progresses, the architectural details will be regularly updated to reflect new insights, advancements, and feedback received during implementation phases. This iterative process will ensure that the framework remains relevant, effective, and aligned with evolving project goals and user requirements.

In conclusion, Deliverable D3.2 lays the groundwork for the successful implementation of the FALCON system. The deliverable is based on the project's Grant Agreement (GAP-101121281) and Consortium Agreement and provides a clear and detailed blueprint that will guide the ongoing development and integration efforts, supporting the project's mission to enhance anti-corruption efforts through advanced technological solutions. This document, also, marks the beginning of a dynamic process that will evolve alongside the FALCON project, ensuring continuous alignment with the project's vision and stakeholder expectations.

9. References

1. Franz, M. et al (2016). Cytoscape.js: a graph theory library for visualisation and analysis: https://academic.oup.com/bioinformatics/article/32/2/309/1744007
2. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083
3. Condessa, F., & Kolter, Z. (2020). Provably robust deep generative models. arXiv preprint arXiv:2004.10608
4. Zhou, D., Wang, N., Gao, X., Han, B., Yu, J., Wang, X., & Liu, T. (2021). Improving white-box robustness of pre-processing defenses via joint adversarial training. arXiv preprint arXiv:2106.05453.
5. Condessa, F., & Kolter, Z. (2020). Provably robust deep generative models. arXiv preprint arXiv:2004.10608.
6. Takemura, T., Yanai, N., & Fujiwara, T. (2020). Model extraction attacks against recurrent neural networks. arXiv preprint arXiv:2002.00123.
7. Tasumi, M., Iwahana, K., Yanai, N., Shishido, K., Shimizu, T., Higuchi, Y., ... & Yajima, J. (2021). First to possess his statistics: Data-free model extraction attack on tabular data. arXiv preprint arXiv:2109.14857.
8. Fredrikson, M., Jha, S., & Ristenpart, T. (2015, October). Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (pp. 1322-1333).
9. Carlini, N., Jagielski, M., & Mironov, I. (2020, August). Cryptanalytic extraction of neural network models. In Annual international cryptology conference (pp. 189-218). Cham: Springer International Publishing
10. Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., ... & Raffel, C. (2021). Extracting training data from large language models. In 30th USENIX Security Symposium (USENIX Security 21) (pp. 2633-2650).
11. Li, Y., Jiang, Y., Li, Z., & Xia, S. T. (2022). Backdoor learning: A survey. IEEE Transactions on Neural Networks and Learning Systems.
12. Subedar, M., Ahuja, N., Krishnan, R., Ndiour, I. J., & Tickoo, O. (2019). Deep probabilistic models to detect data poisoning attacks. arXiv preprint arXiv:1912.01206.