Grant Agreement No.: 101080718
Call: HORIZON-HLTH-2022-STAYHLTH-01-two-stage
Topic: HORIZON-HLTH-2022-STAYHLTH-01-05-two-stage
Type of action: HORIZON-RIA

BIO-STREAMS

D5.1 Technical Architecture

Revision: v.1.0

Work package	WP5
Task	T5.1
Due date	31/10/2024
Submission date	31/10/2024
Deliverable lead	INTRA
Version	0.5
Authors	Theodora Brisimi (INTRA), Anastasios Gogos (INTRA), Marianna Panagiotidou (AIN), Ali Saad (AIN), Elena Politi (HUA), Thelma Androutsou (CSCY), Stavros Pitoglou (CSCY), Nikos Alimpertis (NVCR), Eleni Georga (UOI), Harry Kakoulidis (TMA), Marios Prasinos (TMA), Konstantinos Bromis (ICCS), Ioannis Kouris (ICCS), Matteo Colombo (i2G)
Reviewers	Georgios Roumelas (UNI), Chara Kapsala (UKEMED), Zenia Koti (UKEMED)

Abstract	This deliverable outlines the technical architecture of the BIO-STREAMS platform, providing key updates since Deliverable D2.3, which presented the conceptual and logical views of the system. It introduces new components related to school pilots, observability, and the cost model, and updates the overall conceptual architecture. Following the 4+1 architectural model, the deliverable details the process view, including sequence diagrams for interactions between components, and the deployment view, with a focus on hardware specifications and security measures. Additionally, it covers the platform's CI/CD tools, software development workflow, and the integration methodology, presenting the integration matrix and a detailed time plan for the Alpha, Beta, and Final releases that support the project's pilots.
Keywords	Federated obesity biobank, technical architecture, sequence diagrams, deployment view, continuous integration, continuous deployment, integration plan

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
V0.1	31/05/2024	ToC	Theodora Brisimi (INTRA)
V0.2	21/06/2024	Updated ToC, contributions in all sections	Theodora Brisimi (INTRA), Anastasios Gogos (INTRA), Marianna Panagiotidou (AIN), Ali Saad (AIN), Elena Politi (HUA), Nikos Alimpertis (NVCR), Eleni Georga (UOI), Harry Kakoulidis (TMA), Marios Prasinos (TMA), Ioannis Kouris (ICCS), Konstantinos Bromis (ICCS), Matteo Colombo (i2G)
V0.3	11/10/2024	Deliverable formatting issues, revision of Open API implementation in BIO-STREAMS, revision of integration matrix	Theodora Brisimi (INTRA), Anastasios Gogos (INTRA), Ioannis Kouris (ICCS)
V0.4	18/10/2024	Under internal review	Georgios Roumelas (UNI), Chara Kapsala (UKEMED), Zenia Koti (UKEMED)
V0.5	23/10/2024	Internal review comments addressed	Harry Kakoulidis (TMA), Theodora Brisimi (INTRA), Elena Politi (HUA), Ioannis Kouris (ICCS)
V0.6	30/10/2024	Deliverable ready for submission	

Disclaimer

Copyright notice

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	R	
Dissemination Level		
PU	*Public, fully open, e.g. web*	**x**
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* *R: Document, report (excluding the periodic and final reports)*
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
DATA: Data sets, microdata, etc
DMP: Data management plan
ETHICS: Deliverables related to ethics issues.
SECURITY: Deliverables related to security issues
OTHER: Software, technical diagram, algorithms, models, etc.

Executive summary

This deliverable outlines the technical architecture of the BIO-STREAMS platform, providing a comprehensive update to the conceptual architecture initially presented in Deliverable D2.3. While D2.3 introduced the high-level conceptual and logical views of the platform, this deliverable presents significant enhancements, including updates to the conceptual architecture and the introduction of new components. These new components address key functionalities related to the school pilots, observability, and the cost model, with their respective logical views detailed in this report.

In alignment with the 4+1 architectural model, we present a complete process view, illustrating all operational processes within the BIO-STREAMS platform through detailed sequence diagrams. The deployment view is also covered, along with the hardware specifications, security measures, and infrastructure considerations. Additionally, this deliverable includes an in-depth description of the CI/CD tooling and the software development workflow to be followed. The integration methodology is explained, supported by an integration matrix that details the communication protocols between platform components, and an integration execution time plan, which outlines the contributions of components across three releases—Alpha, Beta, and Final—to meet the needs of the pilot projects.

Table of contents

Contents

List of figures

List of tables

Abbreviations

ACT	(BIO-STREAMS component) ActiveHealth application
AINIGMA	(BIO-STREAMS partner) AINIGMA TECHNOLOGIES
ANA	(BIO-STREAMS component) Analytics service
API	Application Programming Interface
BMI	Body Mass Index
BLOCKS	(BIO-STREAMS partner) BLOKS ZDRAVNI I SOTSIALNI GRIZHI EOOD
BNB	(BIO-STREAMS component) BIO-STREAMS Node Bundle
CD	Continuous Delivery
CHUL	(BIO-STREAMS partner) CENTRE HOSPITALIER UNIVERSITAIRE DE LIEGE
CI	Continuous Integration
CI/CD	Continuous Integration/Continuous Delivery
CLI	Command-Line Interface
CoAP	Constrained Application Protocol
COST	(BIO-STREAMS component) Cost model estimator
CSCY	(BIO-STREAMS partner) CSCY COMPUTER SOLUTIONS CYPRUS LTD
DASH	(BIO-STREAMS component) Dashboard
DevOps	Development and Operations
DoA	Description of action
EIF	European Interoperability Framework
FIWARE	Future Internet-ware
GNU	GNU's Not Unix
HTTP	HyperText Transfer Protocol

HUA	(BIO-STREAMS partner) CHAROKOPEIO PANEPISTIMIO
IAM	Identity and Access Management
ICCS	(BIO-STREAMS partner) EREVNITIKO PANEPISTIMIAKO INSTITOUTO SYSTIMATON EPIKOINONION KAI YPOLGISTON-EMP
ID	Identifier
IIRA	Industrial Internet Reference Architecture
i2G	(BIO-STREAMS partner) I2GROW INNOVATION TO GROW SRL
IMS	(BIO-STREAMS component) Information management system
INTRA	(BIO-STREAMS partner) NETCOMPANY-INTRASOFT SA
IT	Information Technology
IVRA	Industry Value Chain Reference Architecture
JSON	Javascript Object Notation
KI	(BIO-STREAMS partner) KAROLINSKA INSTITUTET
LOGS	(BIO-STREAMS component) Distributed log server
LOINC	Logical Observation Identifiers Names and Codes
MARTEL	(BIO-STREAMS partner) MARTEL GMBH
METRICS	(BIO-STREAMS component) Metrics server
MQTT	Message Queuing Telemetry Transport
MUALC	Multi-Application Unified API for Linked Context
NGSI-LD	Net Generation Service Interface – Linked Data
NGW	(BIO-STREAMS component) Node gateway
NKUA	(BIO-STREAMS partner) ETHNIKO KAI KAPODISTRIAKO PANEPISTIMIO ATHINON
NUCLIO	(BIO-STREAMS partner) NUCLIO NUCLEO INTERACTIVO DE ASTRONOMIA ASSOCIACAO
NVCR	(BIO-STREAMS partner) D.TSAKALIDIS-G.DOMALIS OE
OAS	OpenAPI Specifications

OPENT	(BIO-STREAMS component) Open toolkit
Orion-LD	Orion Linked Data
PENTELI	(BIO-STREAMS partner) GENIKO NOSOKOMEIO PAIDON PENTELIS
RAID	Redundant Array of Independent Disks
RAMI	Reference Architecture Model Industrie
RBAC	Role-Based Access Control
RECENG	(BIO-STREAMS component) Recommendation engine
ReEIF	Refinement of the eHealth European Interoperability Framework
REST API	Representational State Transfer Application Programming Interface
RISKA	(BIO-STREAMS component) Risk assessment tool
SDG	(BIO-STREAMS component) Synthetic data generator
SERG	(BIO-STREAMS component) Serious games suite
SMS	(BIO-STREAMS component) Security monitoring service
SNOMED CT	Systematized Nomenclature of Medicine – Clinical Terms
SSH	Secure Shell
SSO	Single Sign-On
STS	(BIO-STREAMS partner) SPHYNX TECHNOLOGY SOLUTIONS AG
TMA	(BIO-STREAMS partner) TELEMATIC MEDICAL APPLICATIONS EMPORIA KAI ANAPTIXI PROIONTO TILIATRIKIS MONOPROSOPIKI ETAIRIA PERIORISMENIS EYTHINIS
TLS	Transport Layer Security
UFW	Uncomplicated Firewall
UKCM	(BIO-STREAMS partner) Univerzitetni klinicni center Maribor
VHIR	(BIO-STREAMS partner) FUNDACIO HOSPITAL UNIVERSITARI VALL D'HEBRON - INSTITUT DE RECERCA
VM	Virtual Machine
YAML	YAML Ain’t Markup Language

1 Introduction

1.1 Deliverable Scope

Deliverable D5.1 "Technical Architecture" is the result of Task 5.1 within WP5 and serves as a continuation of D2.3 "Specifications & Conceptual Architecture", which was delivered in Month 12 (M12). As we follow the 4+1 architectural model, D2.3 introduced the initial conceptual architecture and the logical view of the system components. In D5.1, we provide significant updates to the conceptual architecture, including the introduction of newly identified components, and present their updated logical views. Additionally, this deliverable introduces the process view, outlining the dynamic interactions between system components, as well as the deployment (physical) view, which details the hardware and infrastructure involved. Lastly, D5.1 covers the integration methodology, the corresponding integration matrix, and the integration time plan. The development activities related to the components described here will be carried out in WP3, WP4, and WP5, with integration activities specifically managed within WP5.

1.2 Deliverable Structure

The rest of the document is structured as follows: Section 2 provides the updates in terms of the conceptual architecture and the logical views of newly introduced components together with their functional and non-functional (system) requirements. Section 3 gives an overview of two reference architecture models, namely the FIWARE and the Refinement of the eHealth European Interoperability Framework and describes how BIO-STREAMS architecture complies with them. Also, some initial OpenAPI specifications are provided. Section 4 describes all processes that occur in the operation of the BIO-STREAMS platform, detailing interactions between the components. Each process view is accompanied by a corresponding sequence diagram. Section 5 provides the hardware specifications, the security measures for the cloud infrastructure and the runtime deployment view of the BIO-STREAMS platform. Section 6 documents the CI/CD tooling together with the workflows that will be used for the development of the BIO-STREAMS platform. Section 7 presents the integration methodology together with an integration matrix listing communication between all pairs of components as well as the integration execution time plan, providing details regarding the functionality with which each component will participate in the three releases planned to serve the project's pilots' needs. Lastly, the deliverable concludes with a summary of the key highlights presented in this report.

2 Updates on D2.3: Specifications & Conceptual Architecture

2.1 Updates on Conceptual Architecture

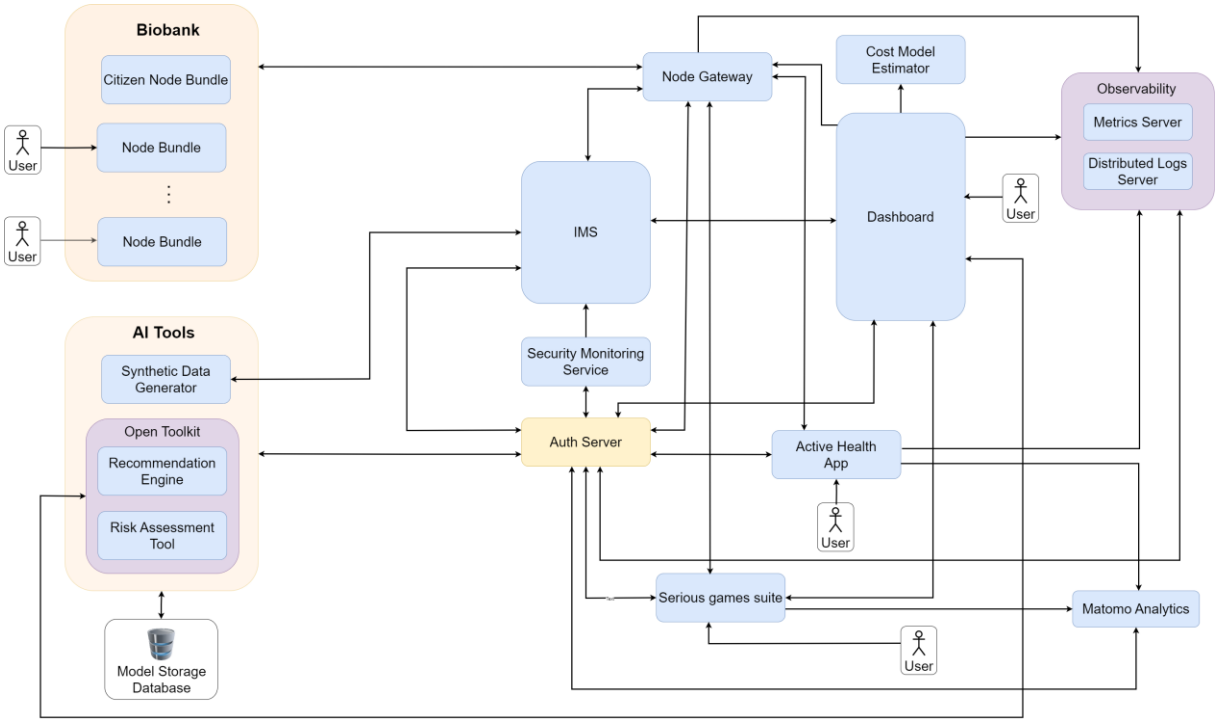

Figure 1: Updated conceptual architecture

This section describes updates in the conceptual architecture as described in previous deliverable D2.3. As a reminder, the main software components presented were the following:

> The **Active Health mobile application (ACT)** and the **Serious Games Suite (SERG)** serve as user-facing interfaces, enabling interaction between end-users and the system. Healthcare professionals use the **Dashboard (DASH)** to monitor and manage end-user data. These interfaces are powered by cloud-based software components, including the **Open Toolkit (OPENT)**, which houses the **Risk Assessment Tool (RISKA)** and the **Recommendation Engine (RECENG)**, as well as the **Synthetic Data Generator (SDG)** and the **Information Management System (IMS)**. Retrospective data flows into the system via **BIO-STREAMS Node Bundles (BNB)**, which are deployed at each clinical pilot site. Finally, a **Security Monitoring Service (SMS)** continuously tracks data interaction logs to ensure the security and compliance of the platform.

A few key changes have been introduced to enhance the system's functionality, particularly in the context of the school pilot data and cost-related analytics. The newly introduced components as well as their description can be found in the following subsections, while the updated conceptual architecture is shown in Figure 1.

2.1.1 New Node Bundle for the School Pilots

A new Node Bundle named **"Citizen Node Bundle**" has been added, to distinct data coming from outside the existing clinical site-specific nodes. This special node bundle is designed to aggregate and store data collected from school pilot programs through the mobile apps (ACT and SERG). The purpose of this new node is to handle data that is not associated with any clinical site, ensuring that school pilot data, which plays a crucial role in our underage obesity studies, is managed appropriately within the system.

2.1.2 New cloud components

Node Gateway (NGW)

The Node Gateway (NGW) will ensure that data generated by the ActiveHealth application, and the Serious Games Suite remains decentralized. Data will not be stored in a central cloud but directly in Node Bundles. Originally outlined in D2.3, the setup used secure SSH connections and a reverse proxy to link Node Bundles with the IMS. Expanding on this, the Node Gateway will be a specific component providing an SSH server and a REST API, enabling connections between cloud services and Node Bundles.

Distributed Log Server (LOGS)

The Distributed Log Server (LOGS) will be used for logging and monitoring events across the BIO-STREAMS platform. It will gather logs from various components such as the Node Gateway, ActiveHealth Application, and Dashboard. This centralized approach will allow technical administrators to search through log data and fix issues efficiently.

Metrics Server (METRICS)

The Metrics Server (METRICS) will collect and display performance data from different components of the BIO-STREAMS platform. It will use Prometheus to gather time-series data and Grafana to create dashboards that visualize this information. This will allow system administrators to monitor important data like server performance, API response times, and database activity.

Analytics Service (ANA)

To track user engagement, behavior, and the use of micro-moments, the BIO-STREAMS platform will use Matomo Analytics, a self-hosted solution. Unlike third-party options like Google Analytics that rely on client-side cookies, Matomo captures behavior from the server side, ensuring full data ownership and privacy. The emphasis on privacy aligns with its use by the European Commission's websites, making it a suitable choice for the platform.

Cost Model Estimator (COST)

The **Cost Model Estimator (COST)**, a vital tool for the BIO-STREAMS project. This component is designed to identify cost-related data within the retrospective datasets being analyzed, leveraging the landscape analysis results from Task 3.2 (reported in D3.1 submitted on August 29th 2024), and existing standards like the European Core Health Indicators. The COST will encode cost-related indicators relevant to underage overweight and obesity, creating a model that links these costs to the broader impact and burden of obesity. In addition, it will propose workflows for structured data collection to ensure the standardized inclusion of cost indicators in the BIO-STREAMS Knowledge Hub, thus providing a comprehensive and effect-based cost analysis framework.

2.2 Logical views of new components

2.2.1 Node Gateway (NGW)

2.2.1.1 Description and Functionality

To preserve the decentralization of data generated by the ActiveHealth application, data, even anonymized, cannot be saved in a central cloud. A mechanism to allow retrieval and storage of the data directly to the Node Bundles is needed.

In D2.3 the architecture to connect the Node Bundles to the IMS was described and was based on a secure SSH connection from each node a central SSH server and maintaining a reverse proxy.

This concept has been expanded and materialized into a specific component called the **Node Gateway (NGW)**. The Node Gateway will provide the SSH server and Rest API to allow connectivity between cloud services and Node Bundles.

- The IMS will connect directly to the API services of the Node Bundles via the Node Gateway.
- The ActiveHealth Application, Serious Games and Dashboard will connect indirectly to the databases of each Node Bundle via the REST API of the Node Gateway.

The Node Gateway via it's REST API, will allow decoupling the various clients (and even 3rd party applications or apps created in the Hackathon, as described in the Description of Action - DoA), from the actual data sources, and fine grain user-level authorization based on the requirements of each application and credentials used by their users.

The security to manage user authentication and access control will be based on OAuth2, with the authentication process based on the central Keycloak server.

The REST API will follow the OpenAPI specification to ensure a clear and consistent way for services to communicate with the Node Gateway. Each endpoint will be precisely defined, outlining the expected request parameters, response formats, and error codes. This will help integrate third-party applications or other services, making it easier for developers to understand how to interact with the Node Gateway and run automated tools that generate client-side code.

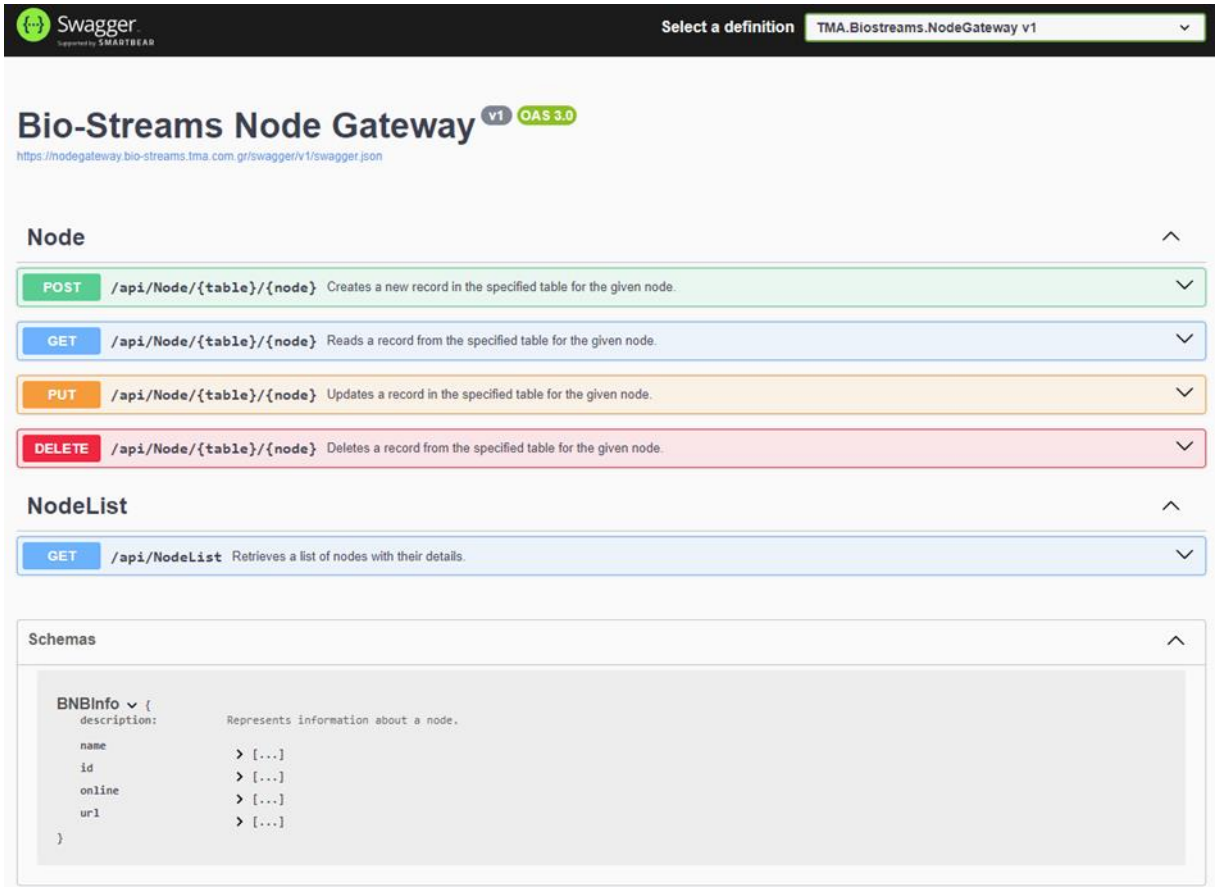

Figure 2: Preliminary Node Gateway OpenAPI specification viewed in the Swagger UI

2.2.1.2 Internal Architecture

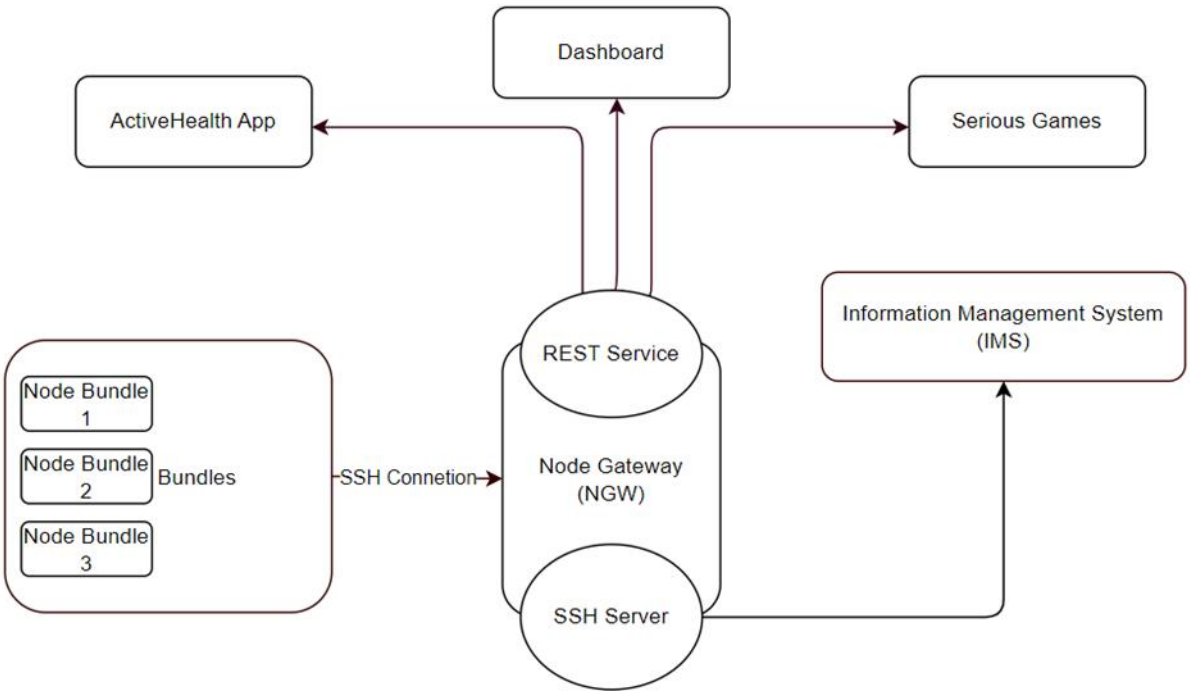

Figure 3: Node gateway internal architecture

2.2.2 Distributed Log Server (LOGS)

2.2.2.1 Description and Functionality

The Distributed Log Server (LOGS) centralizes logging and event monitoring across the BIO-STREAMS platform. It will aggregate logs from multiple components, such as the Node Gateway, ActiveHealth Application, Dashboard and any other service that will implement logging. The component will allow for in-depth analysis and querying of log data, enabling quick identification and resolution of system issues. This will ensure that any irregularities within the platform will be detected and addressed promptly, contributing to smooth and reliable operations.

The LOGS component, powered by **Seq** [1], receives log data via standard HTTP requests, allowing various services to easily send their structured log entries over the network. The HTTP-based ingestion protocol ensures that logs can be sent in a platform-agnostic manner from any component that implements logging, making integration straightforward. Logs are typically formatted using structured logging formats such as JSON, which Seq can parse and index efficiently. This enables system administrators to easily search, filter, and analyze logs across the platform.

To facilitate the implementation of logging, open-source libraries are available for various programming languages that support structured logging and can be configured to send logs directly via HTTP.

The service names used in logging entries follow the naming of the components in this deliverable (e.g. DASH, ACT etc.). For example, in the screenshot below, the DASH component logs a *trace* that contains multiple *spans*. Also, a trace entry from the ACT component is shown.

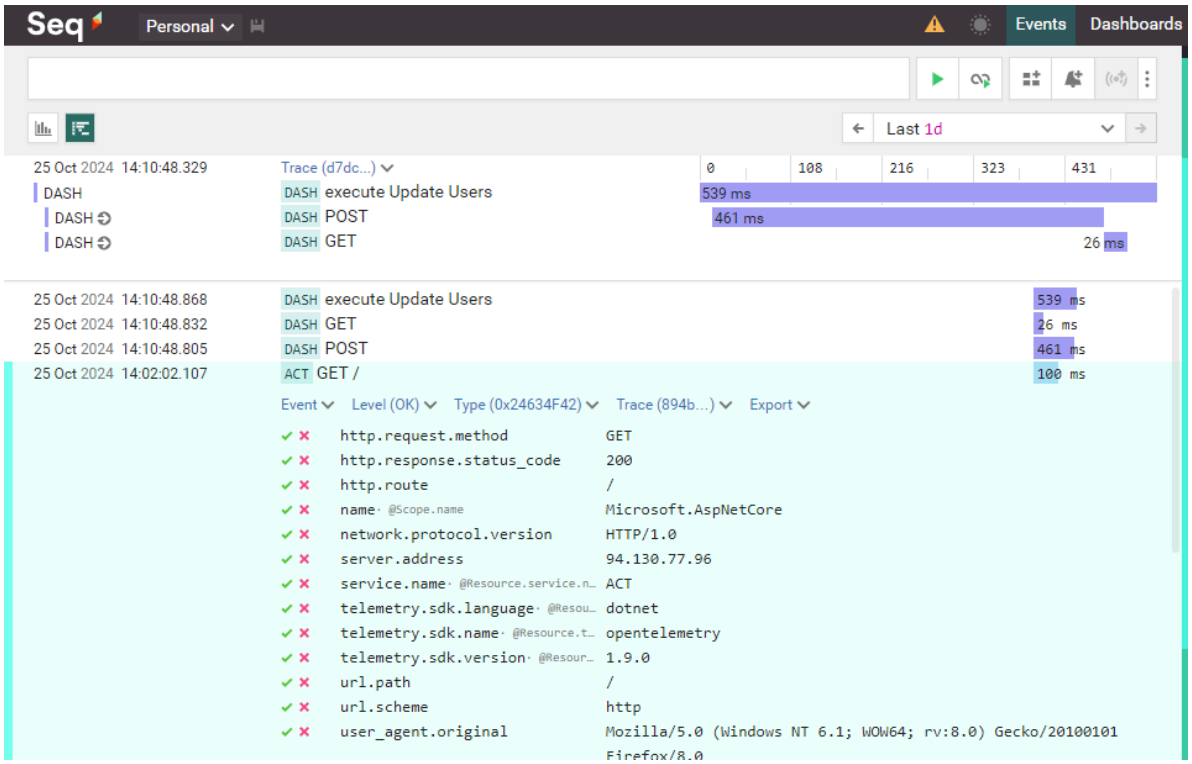

Figure 4: Example of span timings as part of a trace.

Privacy will be preserved by design. Information stored in the logs will exclude any personal identifying details such as usernames and IP addresses, ensuring user privacy protection across the platform.

Open-source tool used: Seq [1]

2.2.2.2 Internal Architecture

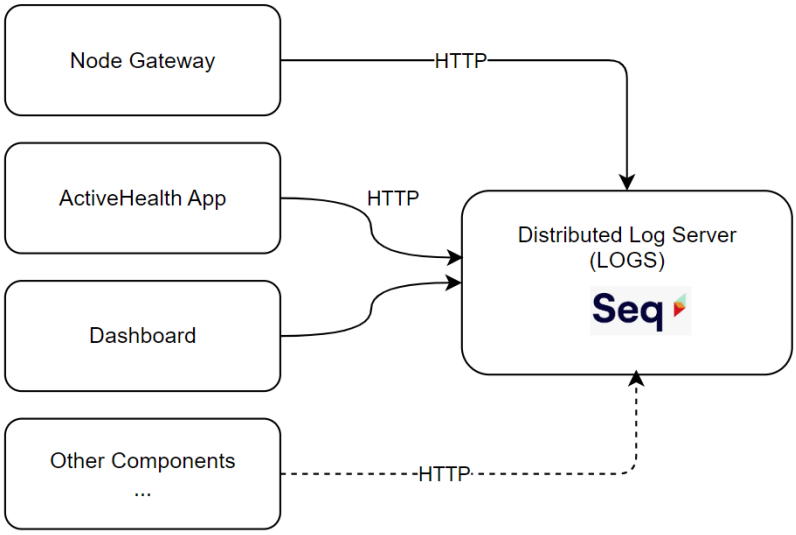

Figure 5: Distributed log server internal architecture

2.2.3 Metrics Server (METRICS)

2.2.3.1 Description and Functionality

The Metrics Server (METRICS) will be responsible for collecting and visualizing performance metrics from components within the BIO-STREAMS platform. Using Prometheus for time-series data collection and Grafana for creating visual dashboards and alerts, the Metrics Server will enable detailed monitoring of various metrics, including server load, API response times, and database performance. This will help ensure the platform's stability and scalability by providing system administrators with the data needed to make informed decisions regarding optimization and performance improvements.

The Dashboard, the ActiveHealth application, and the Node Gateway have been developed to include endpoints that distribute metric data using the OpenTelemetry [2] framework (Figure 5). In the current configuration, Prometheus requests this data from each component at specific intervals via their HTTP metrics endpoint, stores it in its time-series database**,** and Grafana visualizes the data by reading from the Prometheus data source.

Of special interest will be the metrics that have to do with the timings of HTTP requests made from the ActiveHealth application and Serious Games to the Node Gateway, because of the decentralized nature of the requests. Grafana supports an alert system, which can notify administrators of any decrease in the perceived user experience by monitoring specific thresholds in the Prometheus data.

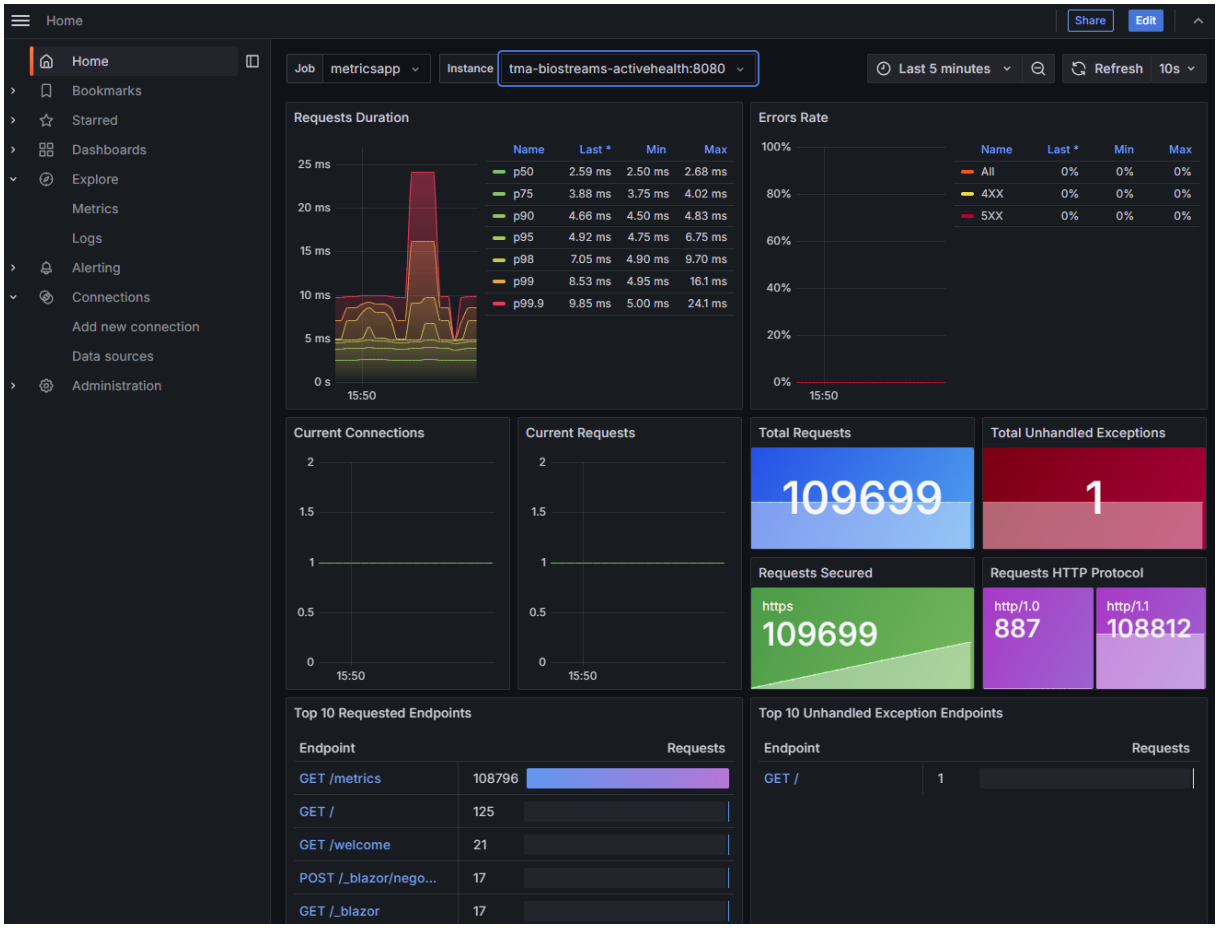

Figure 6: Grafana displaying the metrics dashboard for ActiveHealth (activity from test users)

Privacy will be maintained within the Metrics Server by design. Metrics collected will exclude any personally identifying information, focusing only on performance metrics without storing usernames, IP addresses, or other identifying details.

Open-source tools used: OpenTelemetry [2], Prometheus [3] and Grafana [4].

2.2.3.2 Internal Architecture

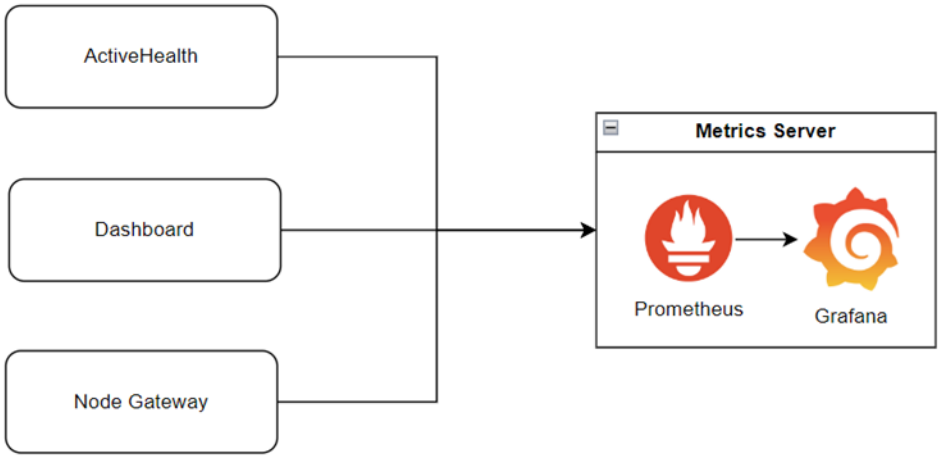

Figure 7: Metrics server internal architecture

2.2.4 Analytics Service (ANA)

2.2.4.1 Description and Functionality

To capture the engagement of the users, their behaviour with the digital interventions, and the use of micro-moments, a self-hosted analytics solution will be used, namely Matomo analytics.

Analytics can provide insights into how users engage with various elements of the app and serious games. For instance, analytics can reveal which features are most engaging, how long users spend on specific views, and whether certain features are underutilized. Analytics can also help track user retention, identifying patterns that encourage users to return to the app.

Using a self-hosted solution that captures the behaviour from the server side (as opposed to third party solutions like Google Analytics with client-side cookies) allows full data ownership and privacy. For these reasons, Matomo is the choice of the European Commission websites themselves.

All the public facing digital intervention tools will send data for analytics:

- The ActiveHealth Application
- The 3 serious games

As the anonymous user ids are shared between these applications, the Analytics Service will be able to track the full experience and behaviour users will have when using the entire spectrum of digital interventions.

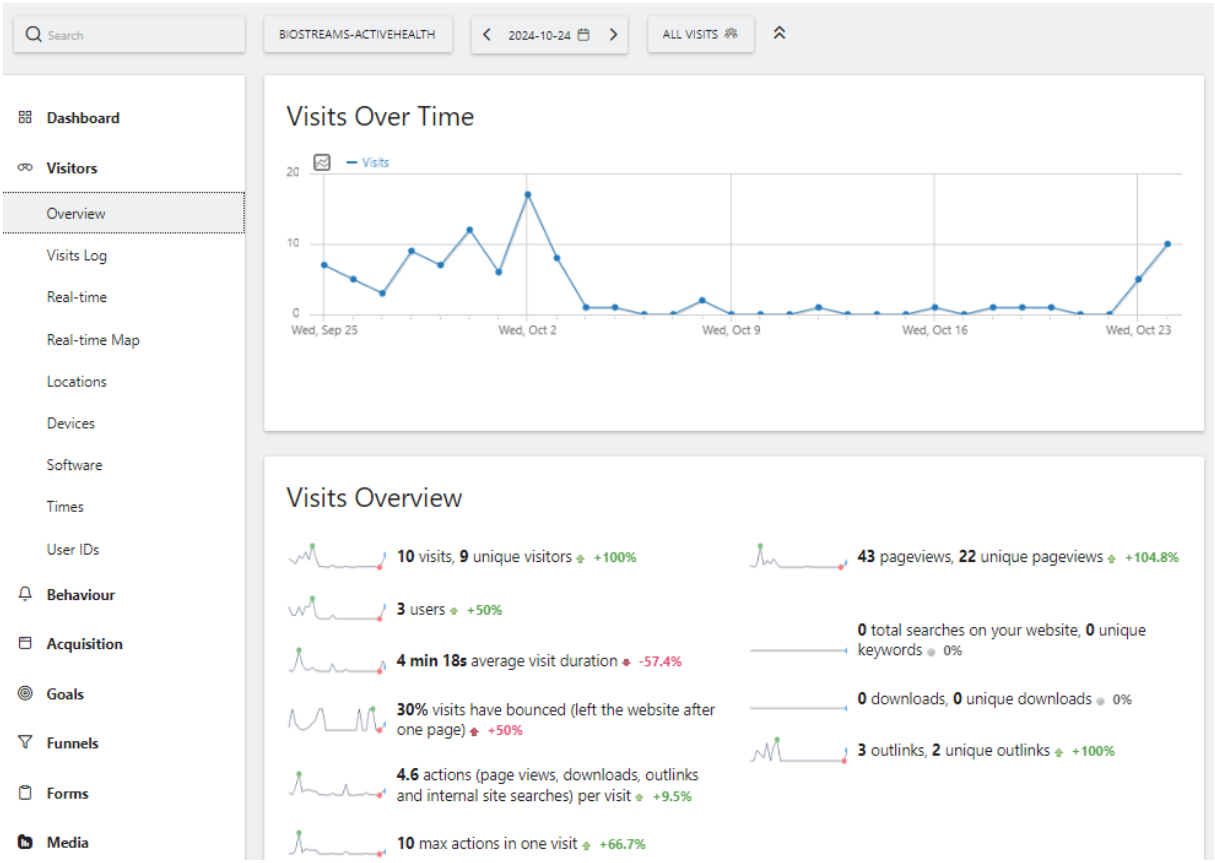

Figure: Example metrics created from activity of test users in the ActiveHealth app

Open-source tools used: Matomo Analytics [5]

2.2.4.2 Internal Architecture

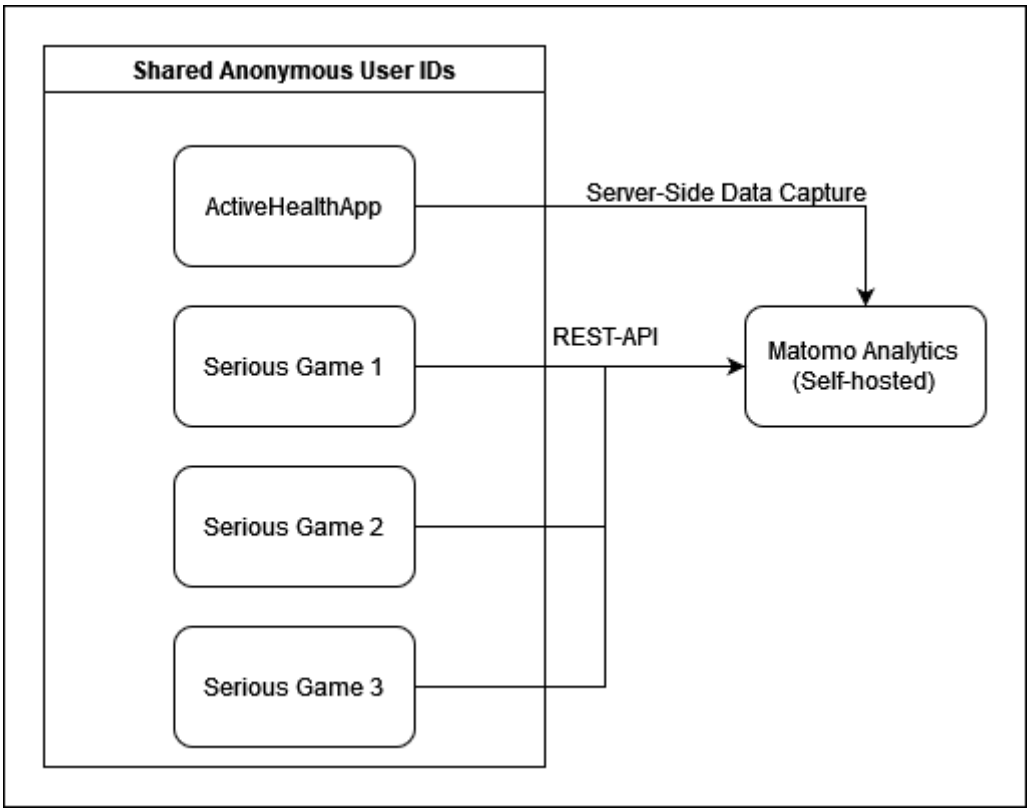

Figure 8: Analytics service internal architecture

2.2.5 Cost Model Estimator (COST)

2.2.5.1 Description and Functionality

The cost model estimator is an application accessible from the BIO-STREAMS Dashboard. The principal end-user of the application are the policy makers (namely health prevention program managers in communities). The application is based on the operationalization of the health economic model developed in Task 3.2 and is based on a workflow management system, APIs to manage the datasets required to use the model and the health economic model to perform simulations related to the economic results of childhood obesity management and prevention programs associating costs and obesity impact/burden, simulating the direct medical costs in the short-term and long-term associated with the programs results based on data provided by the user.

2.2.5.2 Actors

The actors interacting with the cost model estimator are policy makers (health prevention manager in communities) and, on demand through a call function, the healthcare providers in communities that might be involved to collaborate and contribute in populating the socio-economic datasets.

2.2.5.3 Internal Architecture

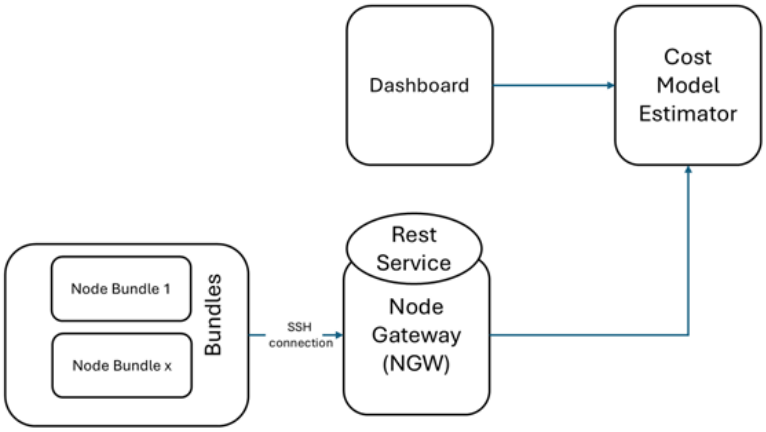

Figure 9: Cost model estimator internal architecture

2.3 Updates on System Requirements

2.3.1 Node Gateway

2.3.1.1 Functional Requirements

Table 1: Node gateway functional requirements

Req #	Func. Req.	Description	Priority	Persona/Associate UR
FR-NGW-01	Secure Data Transfer	The Node Gateway must establish a secure communication channel between Node Bundles and the IMS using SSH.	Must	• Biomedical data scientist/All components: Data Access and Management (#41)

FR-NGW-02	REST API Support	The Node Gateway must provide a REST API for external applications to access data stored in the Node Bundles.	Must	Parent/ActiveHealth mobile app: Monitoring and Tracking of Progress (#31)
FR-NGW-03	Data Routing	The system should route data requests efficiently between Node Bundles and external applications.	Must	• Parent/ActiveHealth mobile app: Monitoring and Tracking of Progress (#31)
FR-NGW-04	Authentication and Authorization	Support authentication for users accessing the REST API and authorize requests based on user roles.	Must	• Parent/ActiveHealth mobile app: Monitoring and Tracking of Progress (#31)
FR-NGW-05	Scalability	The Node Gateway should be capable of handling multiple concurrent connections without performance degradation.	Must	• Parent/ActiveHealth mobile app: Monitoring and Tracking of Progress (#31)

2.3.1.2 Non-Functional Requirements

Table 2: Node gateway non-functional requirements

Req #	Func. Req.	Description	Priority	Persona/Associate UR
NFR-NGW-02	High Availability	The Node Gateway should ensure minimal downtime with failover mechanisms.	Must	Generic not-functional requirement
NFR-NGW-03	Data Encryption	Data transmitted through the Node Gateway should be encrypted using industry-standard protocols like TLS/SSL.	Must	Generic not-functional requirement
NFR-NGW-04	Robust Logging	The Node Gateway should maintain logs of all data transfers and API requests for auditing purposes.	Should	Generic not-functional requirement

NFR-NGW-05	Modular Architecture	The Node Gateway should be designed in a modular way to facilitate easy updates and feature additions.	Should	Generic not-functional requirement

2.3.2 Distributed Log Service

2.3.2.1 Functional Requirements

Table 3: Distributed log server functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
FR-LOGS-01	Centralized Log Aggregation	The Distributed Log Server must aggregate logs from all services, including Node Gateway, ActiveHealth App, and Dashboard.	Must	n/a
FR-LOGS-02	Querying and Filtering	The system should provide an interface to query and filter logs based on various criteria like timestamps and log levels.	Should	n/a
FR-LOGS-03	Log Retention Policy	A configurable retention policy must be supported to manage the storage duration of logs.	Must	n/a
FR-LOGS-04	Integration with Visualization Tools	The server should integrate with visualization tools for real-time log analysis.	Should	n/a
FR-LOGS-05	Secure Access to Logs	Ensure that access to log data is restricted to authorized users only.	Must	n/a

2.3.2.2 Non-Functional Requirements

Table 4:Distributed log server non-functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
NFR-LOGS-01	High Throughput	The system should be capable of processing a large volume of log data without performance degradation.	Must	n/a
NFR-LOGS-02	Fault Tolerance	The Distributed Log Server should continue to operate smoothly even if some components fail.	Must	n/a
NFR-LOGS-03	Data Encryption	Logs should be encrypted in transit to maintain data confidentiality.	Must	n/a
NFR-LOGS-04	Low Latency for Real-Time Analysis	Logs should be available for analysis in near real-time to support quick troubleshooting.	Should	n/a
NFR-LOGS-05	Scalability	The system should be able to scale horizontally to accommodate increased logging volume as the platform grows.	Should	n/a
NFR-LOGS-01	High Throughput	The system should be capable of processing a large volume of log data without performance degradation.	Must	n/a

2.3.3 Metrics Service

2.3.3.1 Functional Requirements

Table 5: Metrics service functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR

FR-METRICS-01	Collect Metrics Data	The Metrics Server must collect metrics data from various services, including Node Gateway, ActiveHealth App, and Dashboard.	Must	n/a
FR-METRICS-02	Store Time-Series Data	It must store collected metrics as time-series data for efficient querying and analysis.	Must	n/a
FR-METRICS-03	Provide Metrics API	The Metrics Server should offer an API for external tools to access collected metrics data.	Should	n/a
FR-METRICS-04	Secure Access to Logs	Ensure that access to log data is restricted to authorized users only.	Must	n/a

2.3.3.2 Non-Functional Requirements

Table 6: Metrics server non-functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
NFR-METRICS-01	High Throughput	The Metrics Server should handle high volumes of metrics data without performance issues.	Must	n/a
NFR-METRICS-02	Data Retention Policy	A configurable data retention policy should be in place to manage the storage duration of time-series data.	Should	n/a

2.3.4 Analytics service

2.3.4.1 Functional Requirements

Table 7: Analytics service functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
FR-ANA-01	Data Collection from Applications	The Analytics Service must collect usage data from the ActiveHealth application and the 3 Serious Games.	Must	n/a
FR-ANA-02	Server-side Data Capture	The system should capture analytics data server-side to ensure full data ownership and avoid reliance on client-side cookies.	Should	n/a
FR-ANA-03	Support for Shared User IDs	The Analytics Service must support tracking of shared anonymous user IDs across different applications to enable analysis of user behavior across tools.	Must	n/a
FR-ANA-04	Data Export and Reporting	The service should provide tools for generating reports and exporting analytics data for further analysis.	Should	n/a

2.3.4.2 Non-Functional Requirements

Table 8: Analytics service non-functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
NFR-ANA-01	Data Privacy and Ownership	The Analytics Service must ensure that all collected data remains fully owned by the hosting entity, with no data being shared with third-party servers.	Must	n/a

NFR-ANA-02	Compliance with Medical Data Standards	The service must comply with privacy regulations, ensuring no leakage of behavioral data, even if anonymized.	Must	n/a
NFR-ANA-03	Data Encryption	Data collected by the service should be encrypted in transit to maintain user privacy.	Must	n/a

2.3.5 Cost Model Estimator

2.3.5.1 Functional Requirements

Table 9: Cost model estimator functional requirements

Req #	Func. Req.	Description	Priority	Persona/Associate UR
FR-COST-01	Data Collection	The Cost Model estimator must access cost-related data (economic and organizational) from the Dashboard and medical and behavioral data from the Node Gateway	MUST	Policy maker/All components: Data and Evidence Access (#44)
FR-COST-02	Reliability	The Cost Model estimator must consistently provide accurate and contextually relevant short- and long-term simulation.	MUST	Policy maker/All components: Data and Evidence Access (#44)
FR-COST-03	Integration with Knowledge Hub	The Cost Model Estimator could be integrated with the Knowledge Hub to provide health economic impact evidences of the BIOSTREAMS intervention/prevention program	COULD	Policy maker/All components: Data and Evidence Access (#44)

2.3.5.2 Non-Functional Requirements

Table 10: Cost model estimator non-functional requirements

Req #	Func. Req.	Description	Priority	Persona/ Associate UR
NFR-COST-01	Data Privacy and Ownership	The Cost Model Estimator must ensure that all collected data remains fully owned by the hosting entity, with no data being shared with third-party servers.	Must	n/a
NFR-COST-01	Compliance with Medical Data Standards	The service must comply with privacy regulations	Must	n/a
NFR-COST-01	Data Encryption	Data collected must be encrypted in transit to maintain user privacy.	Must	n/a

2.3.6 Updates on Functional Requirements of Existing Components

Table 11: Updates on functional requirements of existing components

Req #	Func. Req.	Description	Updates (what changed)
FR-RISKA-01	Anthropometric input features	The RISKA should incorporate anthropometric factors such as the Body Mass Index (BMI), Body Fat Distribution (BFD) & the Mid-Upper Arm Circumference (MUAC) in its analysis for risk level assessment.	Changed to: Such as instead of including in [...] anthropometric factors such as the Body Mass [...]
FR-RISKA-06	Age-specific stratification	The RISKA must be accessed either through the BIO-STREAMS Platform or via the ActiveHealth App.	Added explanations: Beta version through Dashboard Final version (if validated) through app.

FR-RISKA-11	Output	The RISKA output must be a risk-based score for each individual user.	Added explanations: Risk based score in the form of Low-Medium-High risk of developing obesity.
FR-RISKA-12	Output channel	The RISKA must send its output to the Dashboard/Active Health App.	Added explanations: The Beta version will be available only to clinicians through the Dashboard. The final version will be available to all (if validated by clinicians) through the ActiveHealth app as well.
FR-RECENG-08	Anthropometric input features	The RECENG should incorporate anthropometric factors such as the Body Mass Index (BMI), Body Fat Distribution (BFD) & the Mid-Upper Arm circumference (MUAC) in its analysis for exercise/nutrition pathway suggestion.	Changed to: Such as instead of including in [...] anthropometric factors such as the Body Mass [...]
FR-DASH-11	Use in clinical studies	Dashboard must be able to create anonymous user ids for participants in personalized and school studies and synchronize them with the authentication server	Added
FR-SDG-05	Output	The SDG component should return augmented datasets containing a mixture of both real and synthetic data	Instead of returning only synthetic data, the component will output a mixture of real and synthetic data
FR-SDG-06	Evaluation	The SDG component should use a statistical evaluator to evaluate the generated synthetic data	Added
FR-IMS-01	Data Integration	Ability to integrate and manage diverse types of data, including clinical, biological, anthropometric, demographic,	Cost data will not be integrated.

		behavioral, genetic, metabolic, hormonal, and cost data.	
FR-IMS-05	Data Statistics Aggregation	IMS needs to provide the functionality of aggregating statistics information about the data in all Node Bundles. This functionality is needed by the SDG component.	Added

3 Alignment of BIO-STREAMS Platform Architecture with Architecture Reference Models and OpenAPI Specifications

3.1 Architecture Reference Models

3.1.1 FIWARE

3.1.1.1 Layers of a FIWARE-based architecture

FIWARE is an open-source platform designed to accelerate the development of smart applications in various domains, including eHealth. Its architecture is modular and layered, promoting scalability, interoperability, and ease of integration. FIWARE extends beyond traditional reference architectures like RAMI 4.0 (Reference Architecture Model Industrie 4.0), IVRA (Industry Value Chain Reference Architecture), and IIRA (Industrial Internet Reference Architecture) by emphasizing a data-driven approach.

The FIWARE architecture consists of several layers [6]:

1. **Device and Data Acquisition Layer**: This foundational layer encompasses all IoT devices, sensors, and data sources responsible for collecting real-time data. It interfaces with physical devices through IoT agents, data collectors, and gateways, enabling efficient data acquisition from diverse sources.
2. **Communication and IoT Backend Layer**: This layer manages communication protocols and ensures reliable data transmission between devices and the central platform. It supports various communication standards such as MQTT, HTTP, and CoAP through protocol adapters, facilitating interoperability among heterogeneous devices.
3. **Context Management Layer**: Central to the FIWARE architecture, this layer is implemented by the Orion Context Broker. It is responsible for managing and storing context information, allowing for the registration, update, and query of context data in a structured and efficient manner. Recent advancements in the Orion-LD Context Broker have enhanced support for NGSI-LD, enabling improved semantic interoperability and linked data capabilities.
4. **Data Processing and Analysis Layer**: This layer handles data enrichment, complex event processing, and analytics to derive meaningful insights from the collected data. It utilizes data processing engines like Cygnus, Cosmos, and QuantumLeap for handling time-series data and performing advanced analytics.
5. **Application and Service Layer**: This layer hosts the applications and services that provide functionalities to end-users. It includes APIs, web services, and applications built on top of the context data managed by the Context Broker.
6. **Security Layer**: This layer has been significantly enhanced to address emerging cybersecurity challenges. It includes components such as Keyrock Identity Manager, Wilma PEP Proxy, and AuthZForce.
7. **Interface and Visualization Layer**: This layer offers user interfaces and dashboards for interaction and data visualization, including web portals, mobile applications, and graphical user interface tools like WireCloud.
8. **Management and Monitoring Layer**: This layer provides tools for system administration, monitoring, and orchestration, ensuring the smooth operation and maintenance of the platform.

By organizing the architecture into these layers and incorporating data-driven principles, FIWARE provides a structured and modular approach that facilitates the development and integration of applications and promotes interoperability while supports scalability. The data-driven architecture enables the decoupling of industrial processes while securing data ownership, allowing for better control over data flow and integration of various processes.

3.1.1.2 Mapping of FIWARE to the BIO-STREAMS platform

Considering the principles and the implementation of FIWARE architecture, BIO-STREAMS has adopted a number of the proposed layers into the implementation of the BIO-STREAMS platform. In that sense, the “Device and Data Acquisition Layer” is implemented on the BNBs and the ACT, which provide data into the platform from different user perspectives. The “Application and Service Layer” has been used as the basis for the development of Node Gateway and the IMS, which perform the communication between the components and the storage of the generated data. The modules of the platform that comprise the AI Tools, has been designed based on the “Data Processing and Analysis Layer” of FIWARE architecture, which contains the components designed to perform the actions related to the machine learning processes, generating synthetic data and ultimately produce risk assessments and provide recommendations. The “Security Layer” has been implemented in the architecture of BIO-STREAMS using two components, the SMS and the Authentication Server. The combination of these two components using open-source identity and access management provides a secure authentication and authorization solution for the platform. The “Interface and Visualization Layer” has been adopted on the development to the DASH component, used to provide a visual interface to the platform. Finally, the “Management and Monitoring Layer” is realized by the Metrics Server and the Matomo Analytics components of the platform, which provide insights on the platform performance and health, facilitating the administration and maintenance of the platform.

FIWARE Layer	BIO-STREAMS platform
Device and Data Acquisition Layer	BIO-STREAMS Node Bundles Active Health mobile application
Data Processing and Analysis Layer	Synthetic Data Generator Recommendation Engine Risk Assessment Tool
Application and Service Layer	Information Management System Node Gateway
Security Layer	Security Monitoring Service Authentication Server
Interface and Visualization Layer	Dashboard
Management and Monitoring Layer	Metrics Server Matomo Analytics

3.1.2 Refinement of the eHealth European Interoperability Framework (ReEIF)

The Refinement of the eHealth European Interoperability Framework (ReEIF) offers a structured approach to achieving interoperability in eHealth systems. Interoperability in healthcare is crucial for ensuring continuity of care, allowing different healthcare providers and systems to communicate and collaborate effectively [7]. The ReEIF model builds upon the original European Interoperability Framework (EIF) by refining and extending certain layers to address the specific needs of eHealth systems [3]. This inclusive approach is designed to support cross-border healthcare initiatives, with the goal of facilitating the integration of diverse healthcare systems across the European Union.

3.1.2.1 Design layers of an eHealth architecture based on ReEIF principles

The ReEIF model refines the original EIF, which consists of four main levels (Legal, Organizational, Semantic, and Technical), into six distinct layers. This refinement provides a more comprehensive implementation framework for modern healthcare systems [3]:

1. **Legal and Regulatory Layer**: This represents the highest level of the framework and addresses the legal and regulatory requirements governing eHealth services. It ensures compliance with regulations such as data protection laws, patient consent protocols, and cross-border data exchange laws.
2. **Policy Layer**: Derived from the original Organizational level, this layer focuses on contracts and agreements between organizations. It formalizes trust, responsibilities, and governance models, thus fostering effective communication and cooperation within the eHealth ecosystem.
3. **Care Process Layer**: Also derived from the Organizational level, this layer involves the alignment of care processes, resulting in integrated care pathways and shared workflows. It tracks and manages workflow processes, ensuring coordinated and efficient care delivery.
4. **Information Layer**: This layer represents the functional description of data models, data elements, and their linkage to standardized terminologies. It ensures that data exchanged between systems is interpretable and meaningful through the use of standardized terminologies like SNOMED CT and LOINC, as well as ontologies and data models.
5. **Application Layer**: This layer encompasses software applications that provide eHealth services. It involves agreements about how information is imported and exported and ensures that healthcare information systems effectively communicate using standard protocols.
6. **IT Infrastructure Layer**: This layer provides the physical and virtual resources required to run eHealth applications, including communication protocols, networking, hardware resources, and cybersecurity measures. It ensures the secure and efficient operation of healthcare information systems.

This inclusive approach is designed to support cross-border healthcare initiatives, with the goal of facilitating the integration of diverse healthcare systems across the European Union [7]. The ReEIF model aims to support modern healthcare interoperability by providing a structured framework that can adapt to emerging technologies and standards. By maintaining a focus on practical implementation and stakeholder engagement, the ReEIF model has the potential to evolve, incorporating lessons learned from real-world implementations and adapting to new challenges in healthcare technology [7] [8].

3.1.2.2 Mapping of ReEIF to the BIO-STREAMS architecture

As previously mentioned in FIWARE architecture, ReEIF architecture principles have also been adopted on BIO-STREAMS architecture implementation. Some of the ReEIF layers overlap with the layers of the FIWARE, such as the “Information Layer” and the “Application Layer”, although, regarding the Information Layer of ReEIF in BIO-STREAMS platform, it also finds application on the data exchange between the BNBs, which implement the Common Health Data Ontology. The “Legal and Regulatory Layer” is implemented by the BNBs, the AI Tools and the SMS, to comply with all the relevant data protection and cross-border data exchange laws. The “IT Infrastructure Layer” enables the platform modules to interact with the rest of the platform, by receiving, storing and providing data when requested. In the BIO-STREAMS platform, the Infrastructure Layer consists of all the required hardware components to host the Virtual Machines where the platform components and the CI/CD environments are hosted (more details in section 5).

ReEIF Layer	BIO-STREAMS platform
Legal and Regulatory Layer	BIO-STREAMS Node Bundles Synthetic Data Generator Recommendation Engine Risk Assessment Tool Security Monitoring Service Authentication Server
Care Process Layer	Information Management System Node Gateway
Application Layer	Dashboard Active Health mobile application
IT Infrastructure Layer	Hosting infrastructure

3.2 OpenAPI

3.2.1 The OpenAPI specification

The OpenAPI Specification (OAS) has become an essential tool in the realm of API development and management. It offers a standardized approach to describing RESTful APIs, bridging the gap between API providers and consumers by facilitating clearer understanding and interaction [9].

At its foundation, OAS provides a method for documenting APIs in a format that is accessible to both humans and machines. This dual-purpose nature allows developers to quickly

comprehend an API's capabilities while also enabling automated processes such as code generation and testing [10].

The specification encompasses a comprehensive set of API details. It includes endpoint definitions and their associated operations, along with the parameters required for each operation. Expected responses and their formats are also documented, as well as authentication methods. Additionally, the specification supports the provision of metadata about the API, such as contact information and licensing details.

One of the notable strengths of OAS is its flexibility in representation. Developers have the option to write their API specifications in either YAML or JSON, allowing them to choose the format that best suits their preferences and project requirements [10].

The adoption of OAS brings numerous advantages to the API development process. It promotes a standardized approach to API design, which improves consistency across different projects and teams. The specification serves as living documentation, reducing discrepancies between the API description and its actual implementation. It enables the automatic generation of both server-side and client-side code in various programming languages, streamlining the development process.

Furthermore, OAS facilitates API testing by allowing the creation of mock servers and test cases based on the specification. It encourages an API-first design philosophy, where the API contract is established before implementation begins. The specification also supports a rich ecosystem of tools for tasks such as UI generation, validation, and more, enhancing the overall development experience.

The most recent major version, OpenAPI 3.0, introduced enhancements to support more complex API scenarios. These improvements include better descriptions for callbacks and links between operations, allowing for more sophisticated API designs [11].

By providing a common language for API description, the OpenAPI Specification has become an invaluable asset in the modern landscape of web service development and integration. It simplifies communication between different stakeholders in API projects and streamlines many aspects of the API lifecycle, from design and development to testing and documentation.

3.2.2 Implementation of the OpenAPI in BIO-STREAMS

The OpenAPI implementation on BIO-STREAMS platform, will allow the interoperability with third-party tools to enable the exchange of information between BNBs and external biobanks and platform modules. The OpenAPI specification provides a standardized way to define RESTful APIs, providing a clear structure for API documentation, client generation and server implementation. The definition is organized into several key sections, each serving a specific purpose.

Using the OpenAPI specification, the platform components will provide a seamless integration with internal and third-party tools, enhancing the interoperability and the standardized communication between the platform components. The aim of the BIO-STREAMS platform architecture is to use industry standards, to allow easy integration with the components and the third-party tools, and in that sense, tools that will be used in the platform have been selected following this strategy. The OpenAPI documents that will be generated will outline API services, independent of programming languages, frameworks, and deployment technologies. These documents may be generated statically or dynamically from an

application. The files that describe the RESTful API in accordance with the OpenAPI specification will be structured as JSON objects and adhere to JSON standards.

The OpenAPI files will provide a comprehensive description of the entire API of the component including:

- API endpoints (e.g. /users)
- Operations for each endpoint (e.g. GET, POST, DELETE)
- Input parameters (e.g. /users?isactive=true)
- Responses and their formats (200: {“name”: “test”})
- Authentication methods

Focusing on the interoperability of the BIO-STREAMS platform, which refers to the capacity of different systems or components to exchange data and utilize information effectively, the development of the components is performed with focus on API interoperability, by incorporating metadata that describes the data and connecting it to a commonly accepted vocabulary.

Currently, Keycloak and Matomo Analytics components implement the OpenAPI specification, while for the rest of the platform components, the consortium aims to adopt the OpenAPI specification wherever is possible, both for intra BIO-STREAMS communication and with third-party tools. As development activities of the BIO-STREAMS platform components are ongoing, the platform components which will expose APIs to third-party services, will be defined during the activities of WP5. This will take place before the release of the Beta version of the platform (M36) for those that will implement the OpenAPI specification. The related authentication and authorization mechanisms for accessing the APIs will also be provide along with the relase of the OpenAPIs (e.g. apikey, OAuth2, etc). The detailed documentation of the OpenAPI specification for the Beta version will be reported in D5.3 and the final OpenAPI documentation with be included in D5.4, along with the Final version of the platform.

4 BIO-STREAMS Platform Process View

The *Process View* in the 4+1 architecture model focuses on the dynamic aspects of the system, detailing how different components interact at runtime to achieve specific business goals or functionalities. It outlines the sequence of operations, concurrent processing, communication patterns, and synchronization between components, ensuring that the system can handle various tasks efficiently. The Process View provides a clear understanding of the internal behavior of the system, highlighting performance, scalability, and coordination between components.

In the table below (Table 12), we outline the key process flows within the BIO-STREAMS platform, detailing the interactions between its components. Each process flow will be represented with its corresponding sequence diagram, illustrating how data and operations flow between components.

Table 12: Listing of processes and involved components in the BIO-STREAMS platform

Section	PF#	Title	Involved Components
4.1	PF1	Data Collection in Node Bundles in Clinical Sites	BNB
4.2	PF2	Clinical Study User Registration	DASH, AUTH
4.3	PF3	ActiveHealth App User Login	ACT, AUTH
4.4	PF4	Prediction of Child Obesity Risk	DASH, RISKA
4.5	PF5	Delivery of Recommendations for Healthy Living based on Guidelines to Healthcare Professionals	DASH, RECENG
4.6	PF6	Delivery of Recommendations for Healthy Living based on Guidelines to ActiveHealth App Users	ACT, RECENG, DASH
4.7	PF7	Generation of Synthetic Data	DASH, IMS, SDG, AUTH
4.8	PF8	Generation of Enriched Synthetic & Real Data	DASH, IMS, BNB, SDG, AUTH
4.9	PF9	Retrieval of Real Data from Node Bundles	DASH, IMS, BNB, AUTH
4.10	PF10	Retrieval of Data Catalogs from Node Bundles	DASH, IMS, BNB, AUTH
4.11	PF11	User Interaction with Knowledge Hub, Community Network and Marketplace	ACT, Public-facing websites

4.12	PF12	User Interaction with the Associative Catalog from within the ActiveHealth Application	DASH, ACT
4.13	PF13	User Interaction with Serious Game	DASH, AUTH, SERG, BNB
4.14	PF14	Participant in Clinical Studies Opt-out	ACT, DASH
4.15	PF15	Cost-related Data Selection	DASH, COST
4.16	PF16	Cost-related Short- and Long-term Simulation	DASH, COST

4.1 Data Collection in Node Bundles in Clinical Sites

Table 13 PF1: Data Collection in Node Bundles in Clinical Sites

ID	**PF1**	
Title	Data Collection in Node Bundles in Clinical Sites	
Involved Components	BNB	
Actors	Authorised user in clinical premises	
Description	Authorised users upload their datasets on the Node Bundles that are installed in each clinical site. The data are processed through the harmonisation, curation and pseudonymisation components that are installed in the backend component of BNBs. The resulted dataset is securely stored locally in the database (MongoDB) of the BNB.	
Trigger	Authorised users initiate the upload of datasets to the BNB.	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	An authorised user logs in to the system using their credentials, gaining access to the Node Bundle installed at the clinical site.
	2	The user selects the relevant dataset to upload from the local system or internal data repository.
	3	The selected dataset is uploaded to the Node Bundle via the frontend interface.

	4	Once uploaded, the Node Bundle's backend initiates automatic processing of the dataset. This includes harmonization, curation, and pseudonymization, as defined by the installed components.
	5	After processing, the resulting dataset is securely stored in the MongoDB database within the Node Bundle's local storage environment.
	6	The user is notified that the dataset upload and processing have been successfully completed.
Alternative Flow	**Cause**	The dataset fails to meet the validation requirements during the data processing step, due to issues such as missing or incorrectly formatted data.
	STEP	**DESCRIPTION**
	1	An authorised user logs in to the system using their credentials, gaining access to the Node Bundle installed at the clinical site.
	2	The user selects the relevant dataset to upload from the local system or internal data repository.
	3	The selected dataset is uploaded to the Node Bundle via the frontend interface.
	4	Once uploaded, the Node Bundle's backend initiates automatic processing of the dataset. This includes harmonization, curation, and pseudonymization, as defined by the installed components.
	5	During processing, the system identifies an issue with the dataset (e.g., missing or non-compliant fields), causing the processing to fail.
	6	The user is prompted to correct the dataset, addressing the errors identified during processing.
	7	After making the necessary corrections, the user re-uploads the dataset.
	8	The backend resumes processing the corrected dataset, including validation. If no further issues are detected, the process continues according to the main success scenario.

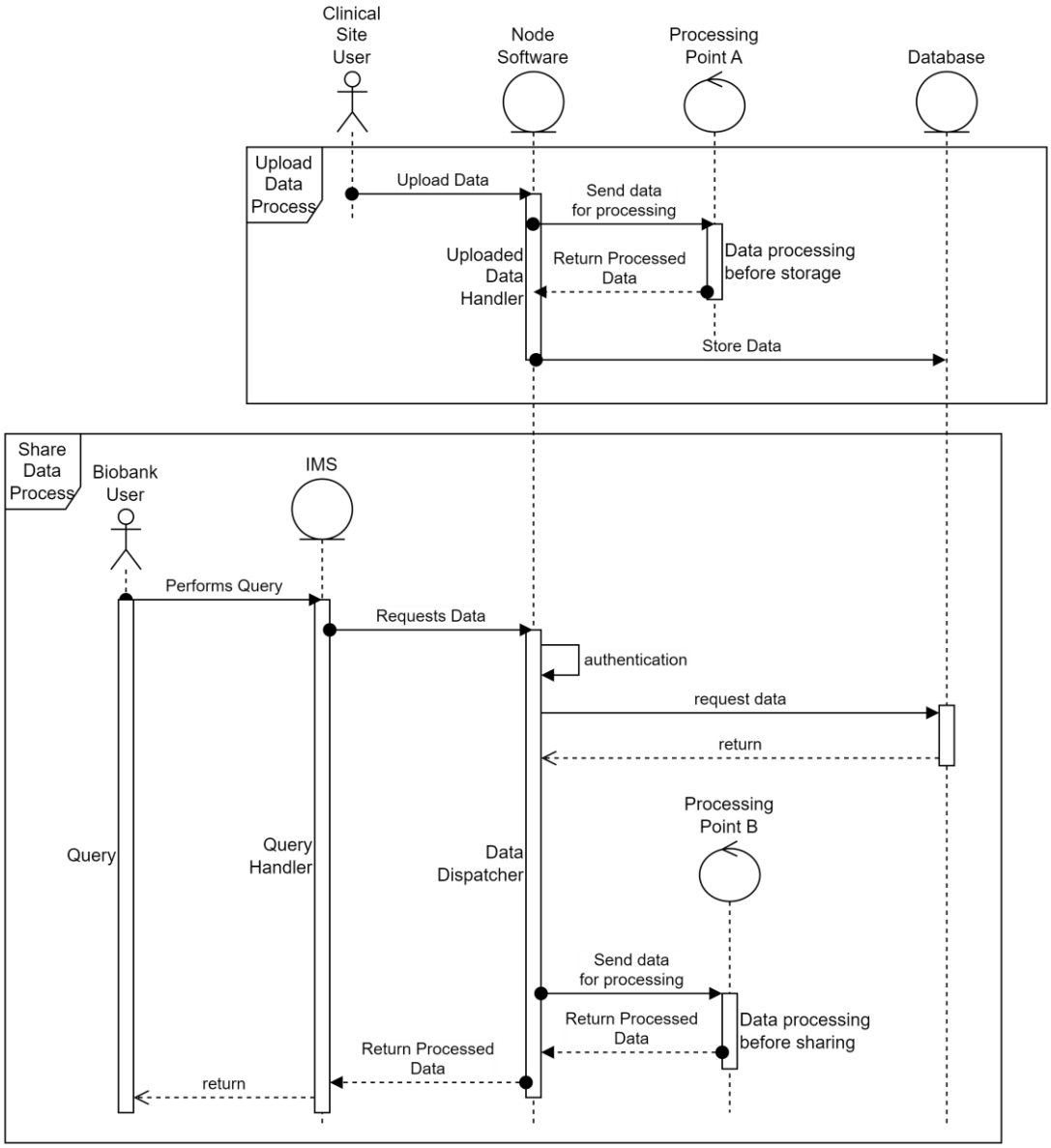

Figure 10: Sequence diagram describing the data collection in Node Bundles in clinical sites

4.2 Clinical Study User Registration

Table 14 PF2: Clinical Study User Registration

ID	PF2
Title	Clinical Study User Registration
Involved Components	DASH, AUTH
Actors	Doctor or Dietitian from a Clinical Site

Description	Creation of the account of new participants in the clinical studies	
Trigger	Navigation to relevant page in Dashboard	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	Health professional clicks the “Register new participant” button.
	2	Random user id is generated, and user created in Dashboard and Keycloak
	3	Printable PDF is generated for the study participant containing id, QR code and directions for the user.
	4	Health professional prints PDF and gives printout to the participant
Alternative Flow	**Cause**	**School Study – Registration of entire classroom**
	STEP	**DESCRIPTION**
	1	School study administrator registers school name, class and number of participants.
	2	Based on the number of participants, random user ids are created
	3	PDF is created with each page corresponding to one participant
	4	Printed pages will be distributed in random to all the class participants.

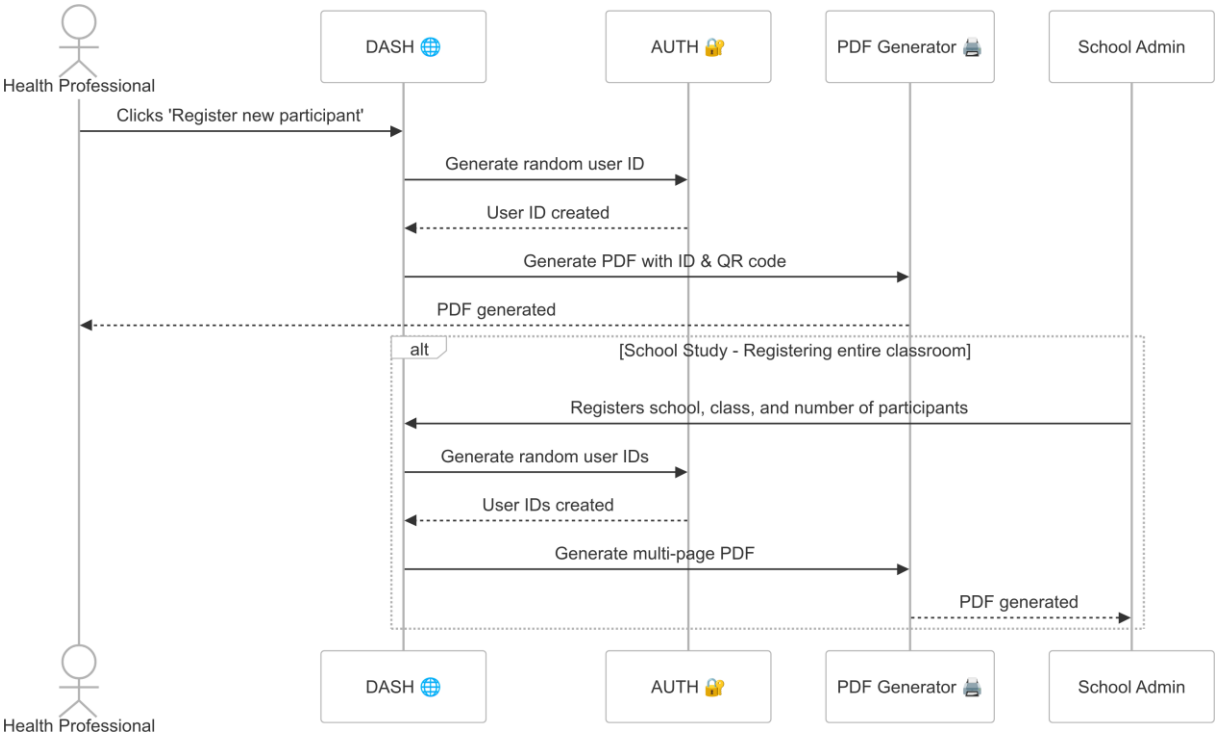

Figure 11: Sequence diagram describing the clinical study user registration

4.3 ActiveHealth App User Login

Table 15 PF3: ActiveHealth App User login

ID	**PF3**	
Title	ActiveHealth App User login	
Involved Components	ACT, AUTH	
Actors	Participants in clinical studies	
Description	User users their user id to log-in to the ActiveHealth app.	
Trigger	First-time launch of ActiveHealth app	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User scans QR code, or navigates to URL of application
	2	Login screen is displayed.
	3	User types in user id. If logging in for the first time, they will use the temporary password provided.

	4	If the login was for the first time, the user will be requested to enter their own permanent password.
Alternative Flow	**Cause**	**Wrong password**
	STEP	**DESCRIPTION**
	1	User cannot log-in because they forgot the password
	2	User clicks “Forgot password” and enters the user id.
	3	Request is made to clinical site and provider resets the password *
	4	User logins again and sets a new password.

* Note that this step cannot be automated, as the system does not have any personalized information about the user, not even an e-mail (that could be used for a password reset link).

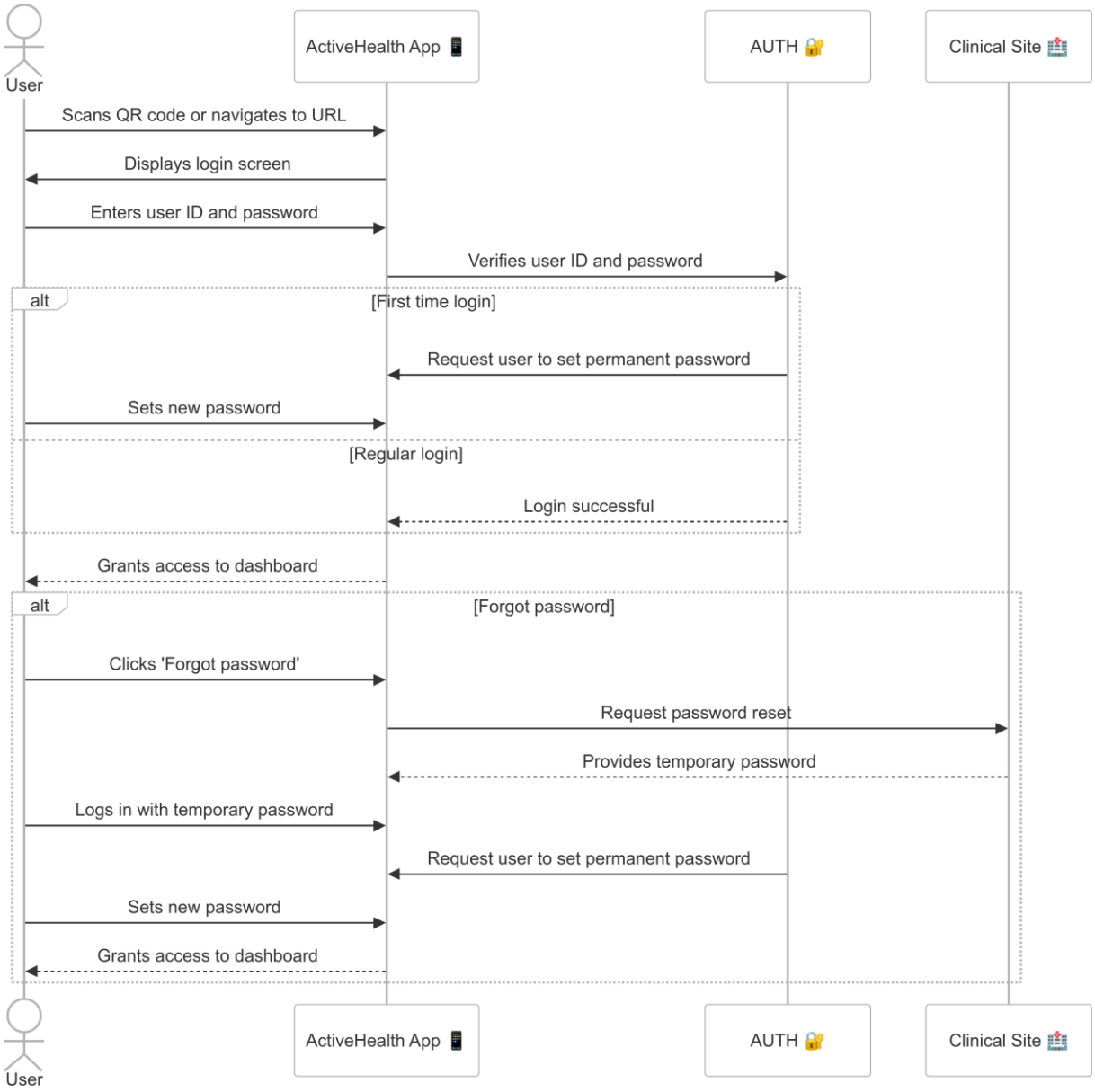

Figure 12: Sequence diagram describing the ActiveHealth app user login

4.4 Prediction of Child Obesity Risk

Table 16 PF4: Prediction of child obesity risk

ID	PF4
Title	Prediction of child obesity risk
Involved Components	DASH, RISKA
Actors	Healthcare Professionals

Description	An authorised and authenticated dashboard user can make use of the risk assessment tool to receive a patient's risk level of developing obesity based on their health data	
Trigger	Initiation action in Dashboard	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	The user initiates the action by selecting the RISKA trigger point in the Dashboard
	2	The user inputs the patient ID
	3	The user views the patient data
	4	The user may update the patient data
	5	The dashboard validates the data format and completeness
	6	The dashboard sends the input data to the RISKA component via REST API
	7	The RISKA component processes the input data into a suitable format
	8	The RISKA component computes a risk score
	9	The RISKA component maps the risk score to a risk level (low, medium, high)
	10	The risk level (and other relevant data) is packaged in JSON format
	11	The response is sent back to the dashboard via REST API
	12	The dashboard receives the output
	13	The dashboard updates the user interface to show the risk level
	14	The user can review the risk level and take appropriate action (confirm, flag, comment)
	15	The user may repeat the process or close the tab.

Alternative Flow	**Cause**	The RISKA Component cannot compute a reliable score because it receives unrealistic values in the input data
	STEP	**DESCRIPTION**
	1	The RISKA component receives input data via the REST API
	2	The RISKA component's internal data validation detects values above or below the set thresholds that correspond to realistic values
	3	A prompt to review the data and correct them is packaged into JSON format
	4	The response is sent back to the dashboard via REST API
	5	The dashboard receives the output
	6	The dashboard updates the user interface and requests the user to update rejected values.

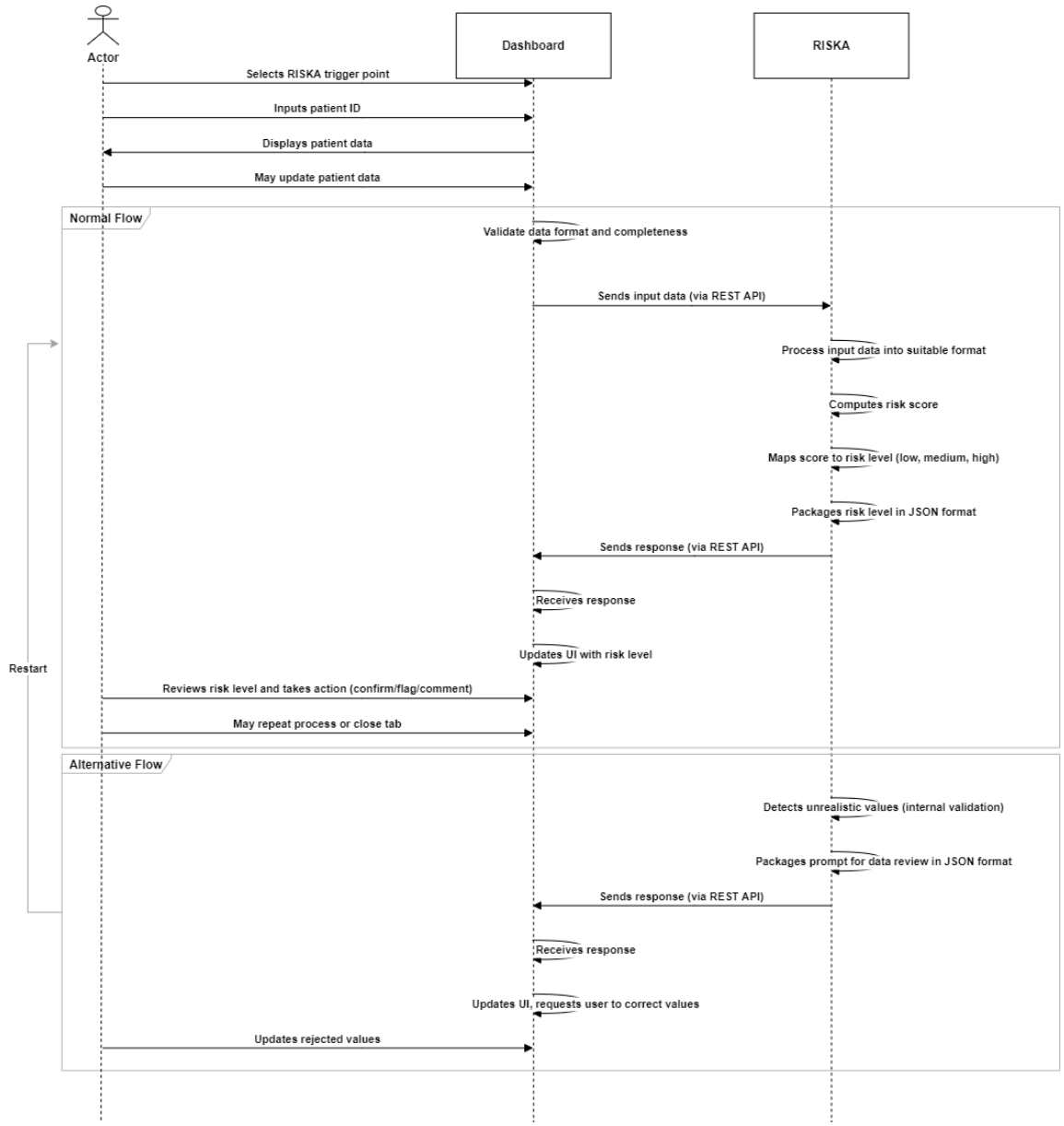

Figure 13: Sequence diagram describing the prediction of child obesity risk

4.5 Delivery of Recommendations for Healthy Living based on Guidelines to Healthcare Professionals

Table 17 PF5: Delivery of recommendations for healthy living based on guidelines to healthcare professionals

ID	PF5
Title	Delivery of recommendations for healthy living based on guidelines to healthcare professionals
Involved Components	RECENG, DASH

Actors	Healthcare professionals	
Description	An authorized and authenticated dashboard user can make use of the recommendation engine to receive recommendations for healthy living for a specific patient based on their health data	
Trigger	Initiation action in Dashboard	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	The user initiates the action by selecting the RECENG trigger point in the dashboard.
	2	The user inputs the patient ID
	3	The user views the patient data
	4	The user may update patient data
	5	The dashboard validates the data format and completeness
	6	The dashboard sends the input data to the RECENG component via REST API
	7	The RECENG component processes the input data into a suitable format
	8	The RECENG component generates the appropriate recommendations
	9	The recommendations (and other relevant data) is packaged in JSON format
	10	The response is sent back to the dashboard via REST API
	11	The dashboard receives the output
	12	The dashboard updates the user interface to show the recommendations
	13	The user can review the recommendation and take appropriate action (confirm, flag, comment)
	14	The user may repeat the process or close the tab

Alternative Flow	**Cause**	The RECENG component cannot generate appropriate personalised recommendations due to insufficient input data
	STEP	**DESCRIPTION**
	1	The RECENG component receives input data with missing non-mandatory values
	2	The RECENG component generates non-personalised recommendations
	3	The recommendations, a message that informs that these recommendations are generic (and other relevant data) is packaged in JSON format
	4	The response is sent back to the dashboard via REST API

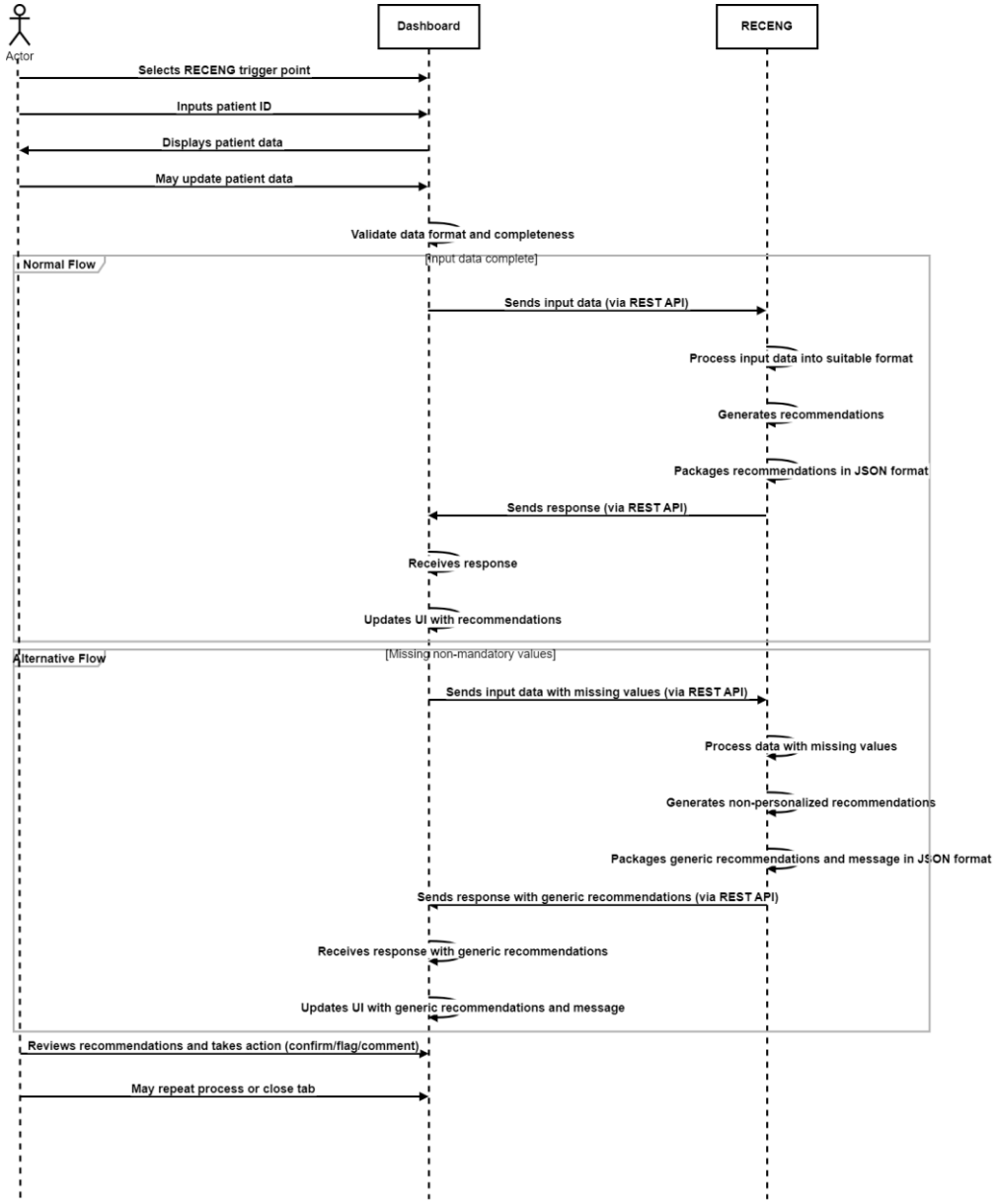

Figure 14: Sequence diagram describing the delivery of recommendations for healthy living based on guidelines to healthcare professionals

4.6 Delivery of Recommendations for Healthy Living based on Guidelines to ActiveHealth App Users

Table 18 PF6: Delivery of recommendations for healthy living based on guidelines to ActiveHealth app users

ID	PF6
Title	Delivery of recommendations for healthy living based on guidelines to ActiveHealth app users
Involved Components	RECENG, ACT, DASH
Actors	Monitors

Description	An authenticated ActiveHealth app users can make use of the recommendation engine to receive recommendations for healthy living based on their health data	
Trigger	Initiation action in ActiveHealth app	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	The user initiates the action by selecting the RECENG trigger point in the ActiveHealth app.
	2	The user views their health data (if any)
	3	The user may add or update health data
	4	The dashboard receives the data from the App
	5	The dashboard validates the data format and completeness
	6	The dashboard retrieves previously generated approved/edited recommendations if any and if the input data haven't been updated.
	7	The dashboard sends the output to the app
	8	The app receives the output and updates the user interface to show the recommendations
	9	The user can review the recommendation and take appropriate action (like, dislike or no response)
	10	The user may repeat the process or close the tab
Alternative Flow	**Cause**	Lack of stored recommendations or Updated input data
	STEP	**DESCRIPTION**
	1	The dashboard cannot retrieve previously generated approved/edited recommendations
	2	The dashboard sends the input data to the RECENG component via REST API
	3	The dashboard receives recommendations from the RECENG component as described in process flow 12.5

	4	The dashboard updates the user interface to show the recommendations

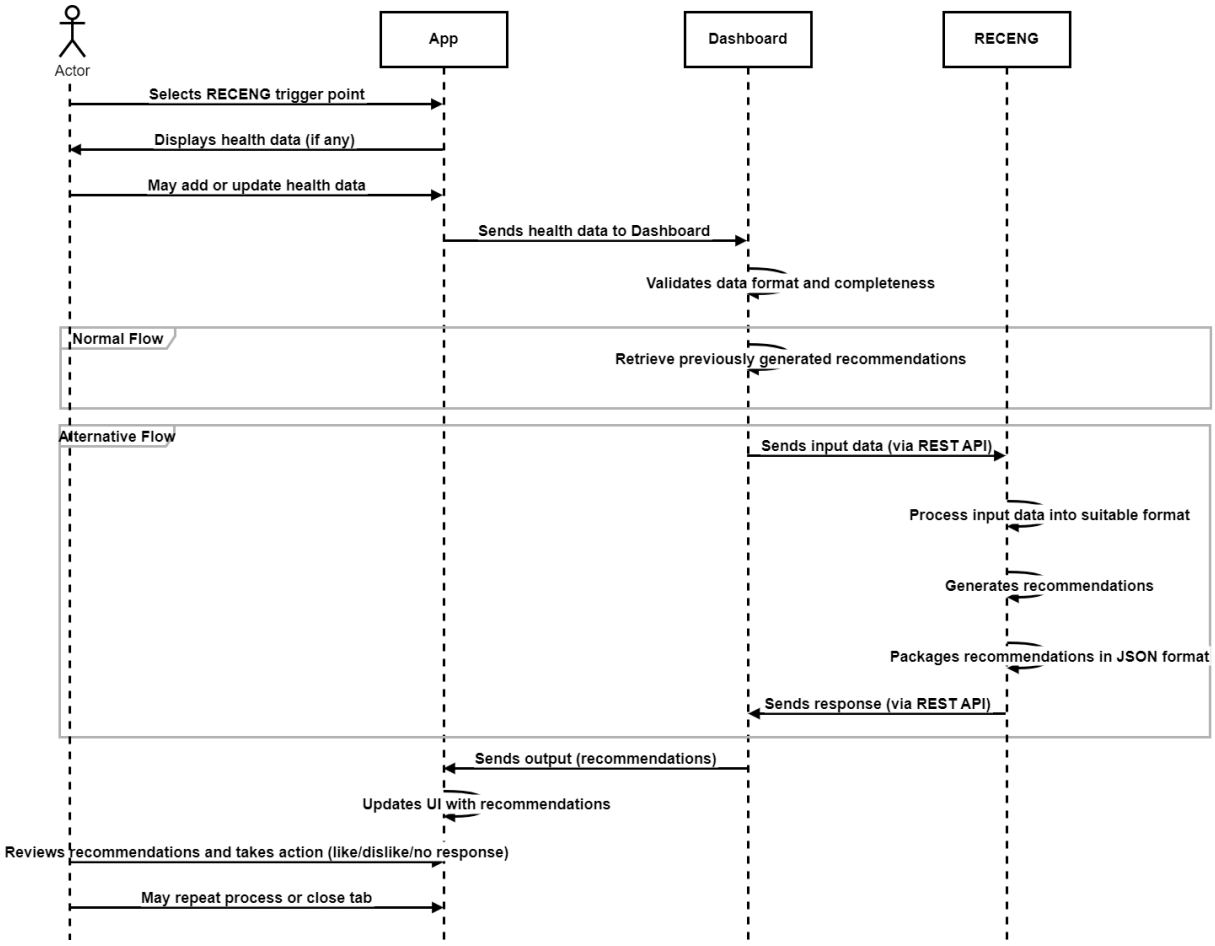

Figure 15: Sequence diagram describing the delivery of recommendations for healthy living based on guidelines to ActiveHealth app users

4.7 Generation of Synthetic Data

Table 19 PF7: Generation of synthetic data

ID	**PF7**
Title	Generation of synthetic data
Involved Components	DASH, AUTH, IMS, SDG
Actors	Authorized dashboard user

Description	Authorized DASH user requests synthetic data to be generated. The data retrieval is taking place through the IMS which applies the role-based access and tracks the actual data transaction	
Trigger	Authorized DASH user visits the relative page and submits a query to request synthetic data	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User visits the Dashboard and performs a successful log-in
	2	User visits the dashboard page that is created for requesting synthetic data and submits a request
	3	The request is forwarded to the IMS, which then forwards the appropriate request to the SDG
	4	SDG generates the synthetic data and responds back to the IMS
	5	IMS creates a log of the transaction that is taking place
	6	IMS replies to the original request (that was submitted by the user) with a payload that contains the data
	7	Dashboard displays the synthetic data to the User
Alternative Flow	**Cause**	Unauthorized request reaches the IMS endpoint. Depending on the role of the user, the Dashboard will provide the appropriate views (pages, menus, etc) by implementing some kind of conditional rendering. In case this is bypassed (e.g. an authenticated, but not authorized, user tries programmatically to access other IMS endpoints with his token) the IMS will respond with “401 Unauthorized” messages when needed.
	STEP	**DESCRIPTION**
	1	Request for retrieving data from the Synthetic Data Generator reaches the IMS.
	2	The IMS checks that it contains a valid token in terms of Authentication (active session of authenticated user)

	3	The IMS identifies that the request comes from an unauthorized user
	4	The IMS replies with a “401 Unauthorized” response and denies access

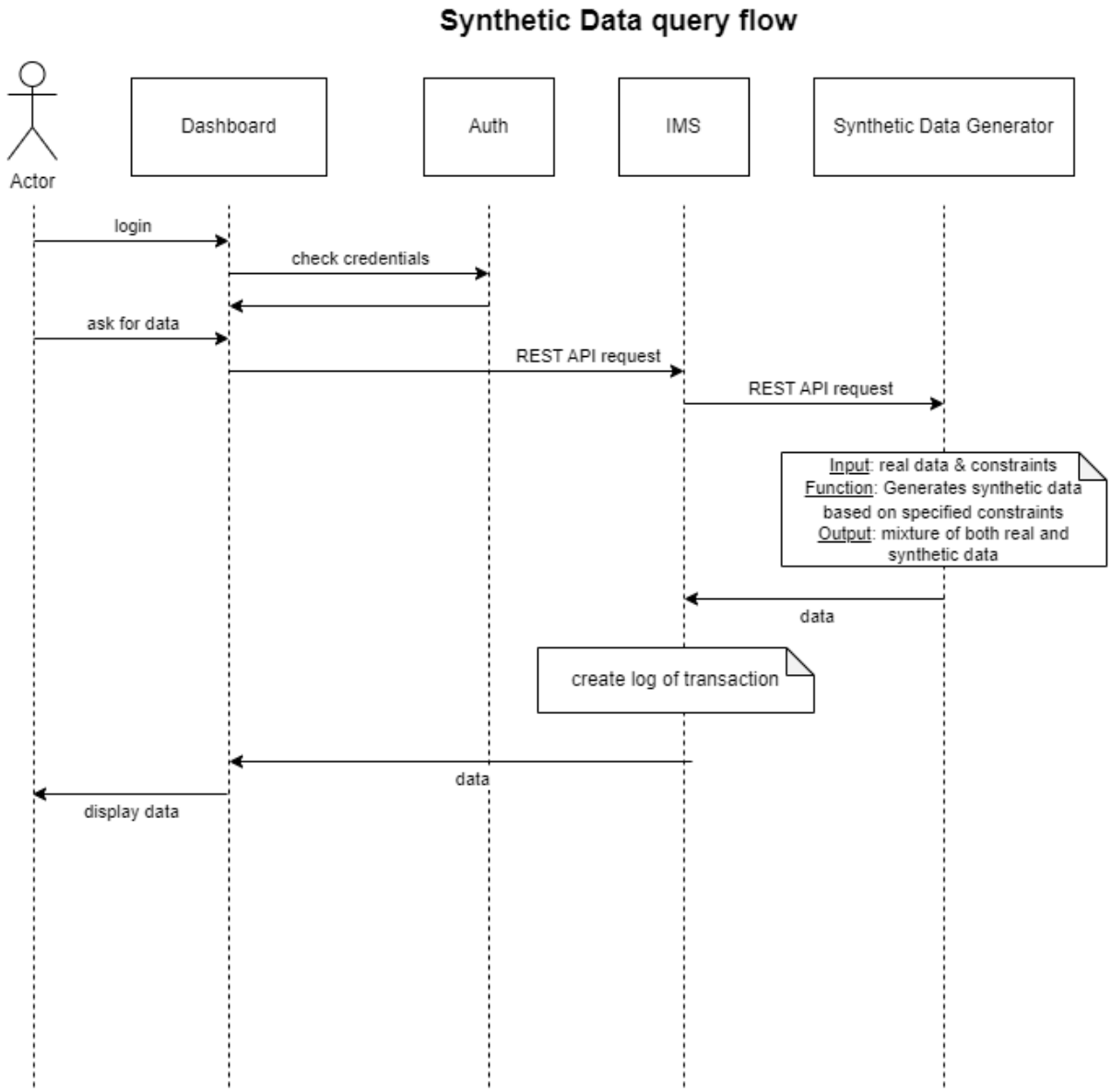

Figure 16: Sequence diagram describing the generation of synthetic data

4.8 Generation of Enriched Synthetic & Real Data

Table 20 PF8: Generation of Enriched Synthetic & Real Data

ID	**PF8**

Title	Generation of Enriched Synthetic & Real Data	
Involved Components	DASH, AUTH, IMS, SDG, BNB	
Actors	Authorized dashboard user	
Description	Authorized DASH user requests synthetic data to be generated. The data retrieval is taking place through the IMS which applies the role-based access and tracks the actual data transaction	
Trigger	Authorized DASH user visits the relative page and submits a query to request Enriched Synthetic & Real Data	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User visits the Dashboard and performs a successful log-in
	2	User visits the dashboard page that is created for requesting Enriched Synthetic & Real Data and submits a request
	3	The request is forwarded to the IMS, which then targets the BNBs needed and forwards them the appropriate requests via the Node Gateway (a communication with more than one -or even all BNBs- is possible at this stage)
	4	NGW forwards the request(s) to the BNB(s)
	5	Targeted BNB(s) retrieve the real data from their database, perform harmonization, pseudonymization, curation and respond back to the NGW
	6	NGW responds back to the IMS with the data from the BNB(s)
	7	IMS requests metadata from the targeted BNB(s) via the Node Gateway
	8	IMS performs a metadata aggregation and sends a request to the SDG including the real, processed data and the aggregated metadata
	9	SDG replies with the enriched, synthetic and real data

	10	IMS creates a log of the transaction that is taking place
	11	IMS replies to the original request (that was submitted by the user) with a payload that contains the data
	12	Dashboard displays the enriched, synthetic and real data
Alternative Flow	**Cause**	Unauthorized request reaches the IMS endpoint. Depending on the role of the user, the Dashboard will provide the appropriate views (pages, menus, etc) by implementing some kind of conditional rendering. In case this is bypassed (e.g. an authenticated, but not authorized, user tries programmatically to access other IMS endpoints with his token) the IMS will respond with “401 Unauthorized” messages when needed.
	STEP	**DESCRIPTION**
	1	Request for retrieving data from the Synthetic Data Generator reaches the IMS.
	2	The IMS checks that it contains a valid token in terms of Authentication (active session of authenticated user)
	3	The IMS identifies that the request comes from an unauthorized user
	4	The IMS replies with a “401 Unauthorized” response and denies access

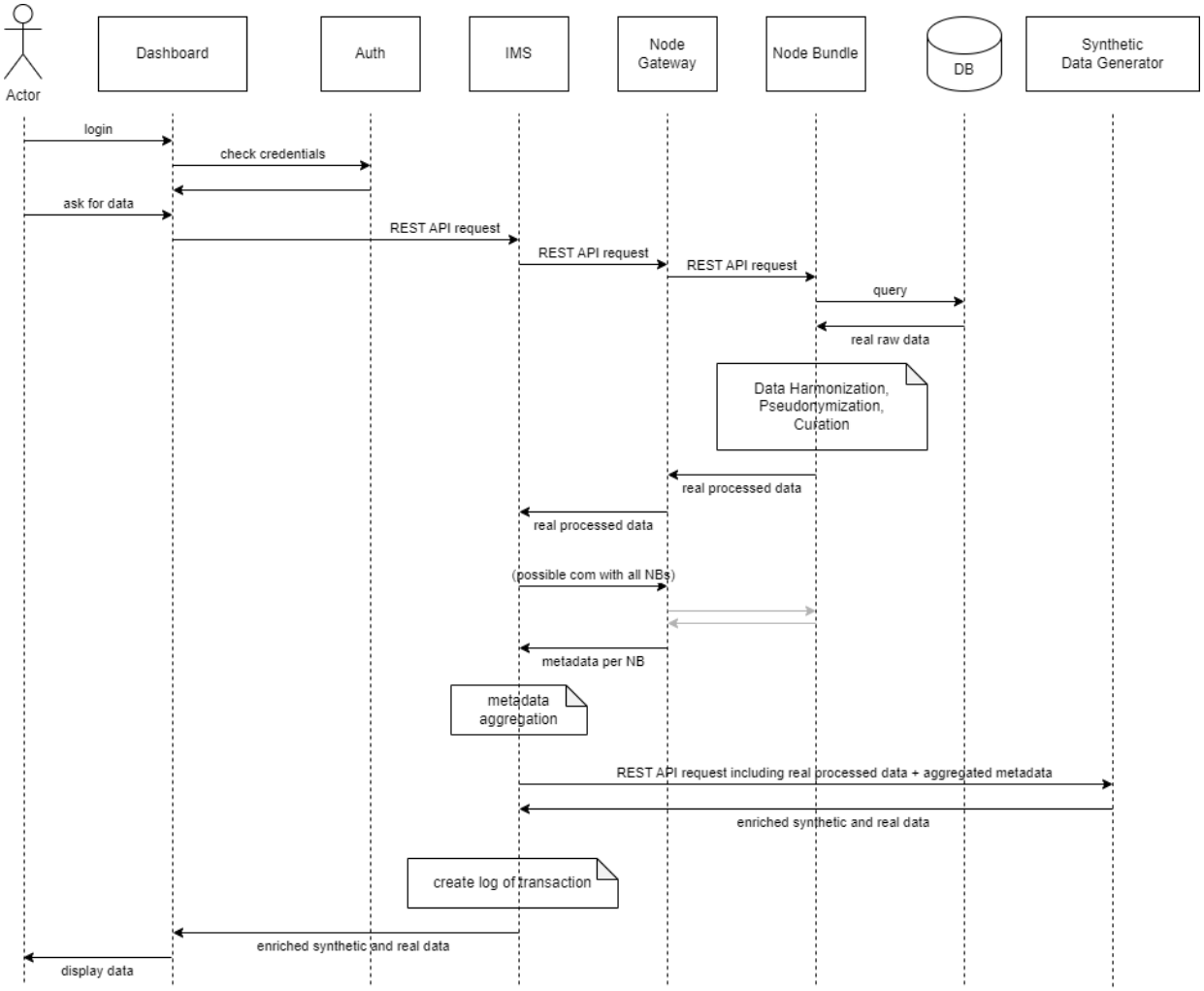

Figure 17: Sequence diagram describing the generation of enriched synthetic & real data

4.9 Retrieval of Real Data from Node Bundles

Table 21 PF9: Retrieval of real data from Node Bundles

ID	PF9
Title	Retrieval of real data from Node Bundles
Involved Components	DASH, AUTH, IMS, BNB
Actors	Authorized dashboard user
Description	Authorized DASH user requests clinical data which are stored in the BNBs. The data retrieval is taking place through the IMS which applies the role-based access and tracks the actual data transaction
Trigger	Authorized DASH user visits the relative page and submits a query to request clinical data

Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User visits the Dashboard and performs a successful log-in
	2	User visits the dashboard page that is created for requesting clinical data and submits a request
	3	The request is forwarded to the IMS which forwards the appropriate request to the Node Gateway
	4	NGW identifies the target BNB and makes the data request
	5	BNB performs a harmonization / pseudonymization / curation of data before they are returned to the IMS
	6	IMS creates a log of the transaction that is taking place
	7	IMS replies to the original request (that was submitted by the user) with a payload that contains the data
	8	Dashboard displays the clinical data to the User
Alternative Flow	**Cause**	Unauthorized request reaches the IMS endpoint. Depending on the role of the user, the Dashboard will provide the appropriate views (pages, menus, etc) by implementing some kind of conditional rendering. In case this is bypassed (e.g. an authenticated, but not authorized, user tries programmatically to access other IMS endpoints with his token) the IMS will respond with “401 Unauthorized” messages when needed.
	STEP	**DESCRIPTION**
	1	Request for retrieving data from the Synthetic Data Generator reaches the IMS.
	2	The IMS checks that it contains a valid token in terms of Authentication (active session of authenticated user)
	3	The IMS identifies that the request comes from an unauthorized user
	4	The IMS replies with a “401 Unauthorized” response and denies access

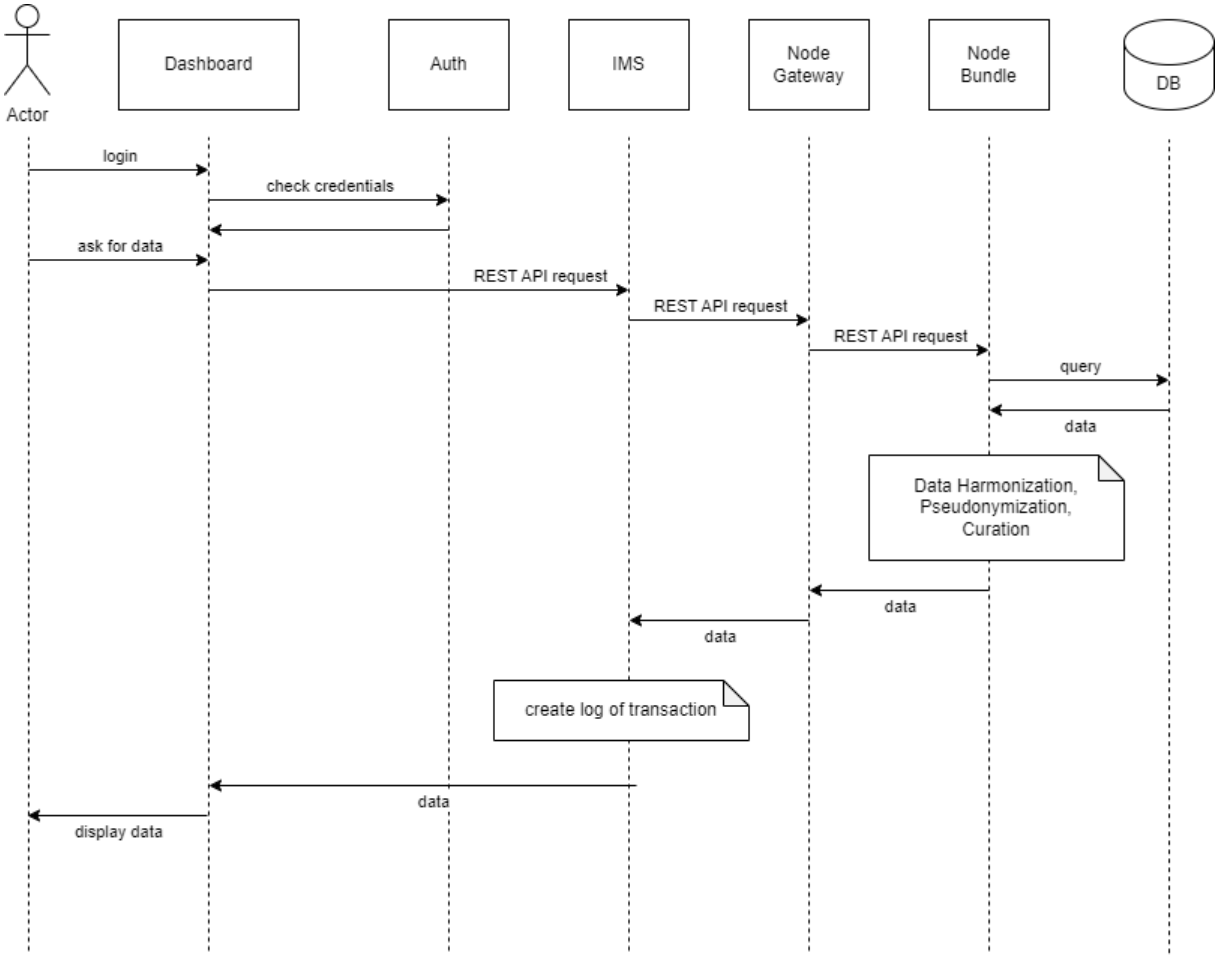

Figure 18: Sequence diagram describing the retrieval of real data from Node Bundles

4.10 Retrieval of Data Catalogue(s) from Node Bundles

Table 22 PF10: Retrieval of data catalogue(s) from Node Bundles

ID	PF10
Title	Retrieval of data catalogue(s) from Node Bundles
Involved Components	DASH, AUTH, IMS, BNB
Actors	Authorized dashboard user
Description	Authorized DASH user requests Data Catalogue from BNB. The data retrieval is taking place through the IMS which applies the role-based access and tracks the actual data transaction
Trigger	Authorized DASH user visits the relative page and submits a query to request the data catalogue

Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User visits the Dashboard and performs a successful log-in
	2	User visits the dashboard page that is created for retrieving the data catalogue(s)
	3	The request is forwarded to the IMS, which identifies the target BNB and forwards the appropriate request to the Node Gateway
		NGW communicates with
	4	BNB responds back to the IMS with the data catalogue(s)
	5	IMS creates a log of the transaction that is taking place
	6	IMS replies to the original request (that was submitted by the user) with a payload that contains the data catalogue(s)
	7	Dashboard displays the synthetic data to the User
Alternative Flow	**Cause**	Unauthorized request reaches the IMS endpoint. Depending on the role of the user, the Dashboard will provide the appropriate views (pages, menus, etc.) by implementing some kind of conditional rendering. In case this is bypassed (e.g. an authenticated, but not authorized, user tries programmatically to access other IMS endpoints with his token) the IMS will respond with “401 Unauthorized” messages when needed.
	STEP	**DESCRIPTION**
	1	Request for retrieving data from the Synthetic Data Generator reaches the IMS.
	2	The IMS checks that it contains a valid token in terms of Authentication (active session of authenticated user)
	3	The IMS identifies that the request comes from an unauthorized user
	4	The IMS replies with a “401 Unauthorized” response and denies access

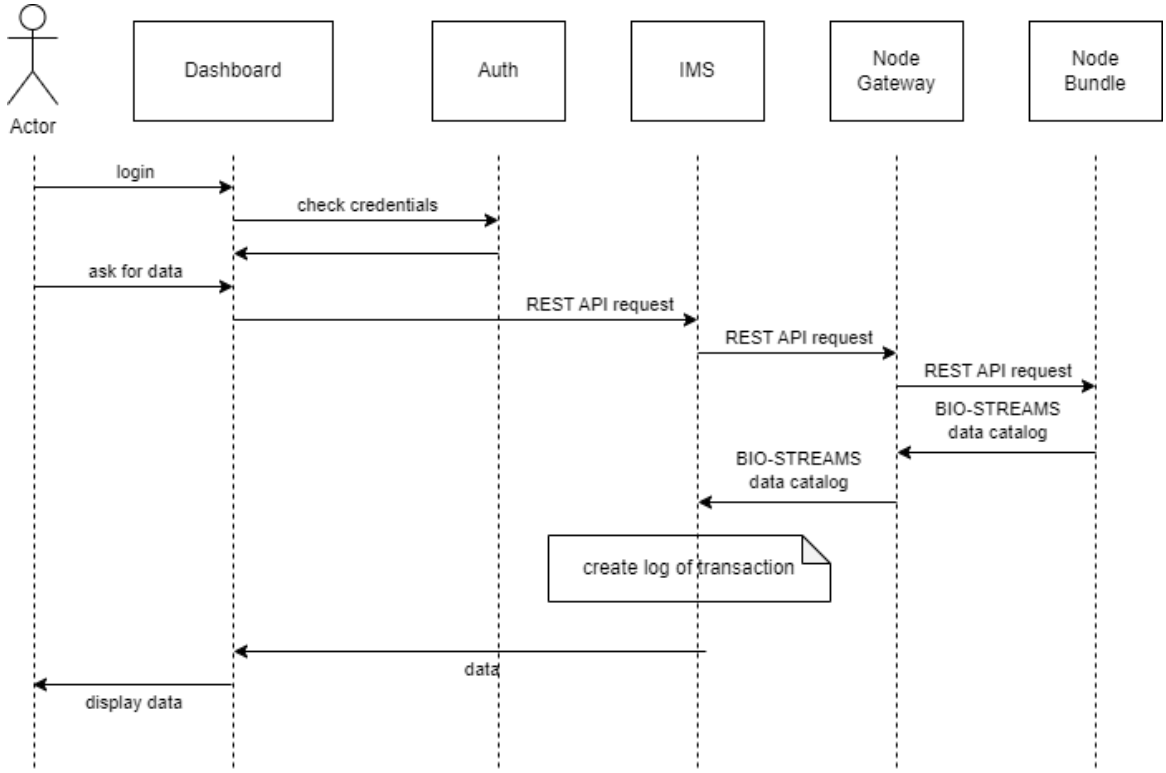

Figure 19: Sequence diagram describing the retrieval of data catalogue(s) from Node Bundles

4.11 User Interaction with Knowledge hub, Community Network and Marketplace

Table 23 PF11: User interaction with Knowledge hub, Community network and Marketplace

ID	**PF11**	
Title	User interaction with Knowledge hub, Community network and Marketplace	
Involved Components	ACT, Public-facing websites	
Actors	Parents	
Description	User wants to view BIO-STREAMS Knowledge hub, Community network or Marketplace	
Trigger	Navigational items in the ActiveHealth Application	
	STEP	**DESCRIPTION**

Main Success Scenario	**1**	User starts the application. If logged in, the app goes directly to the user home page.
	2	Users chooses website from one of the navigational items
	3	Corresponding website is launched
Alternative Flow	**Cause**	**User is not authenticated**
	STEP	**DESCRIPTION**
	1	New users are greeted with a welcome page that contains a slider with various Bio-Stream services.
	2	User chooses website from the homepage slider
	3	Corresponding website is launched

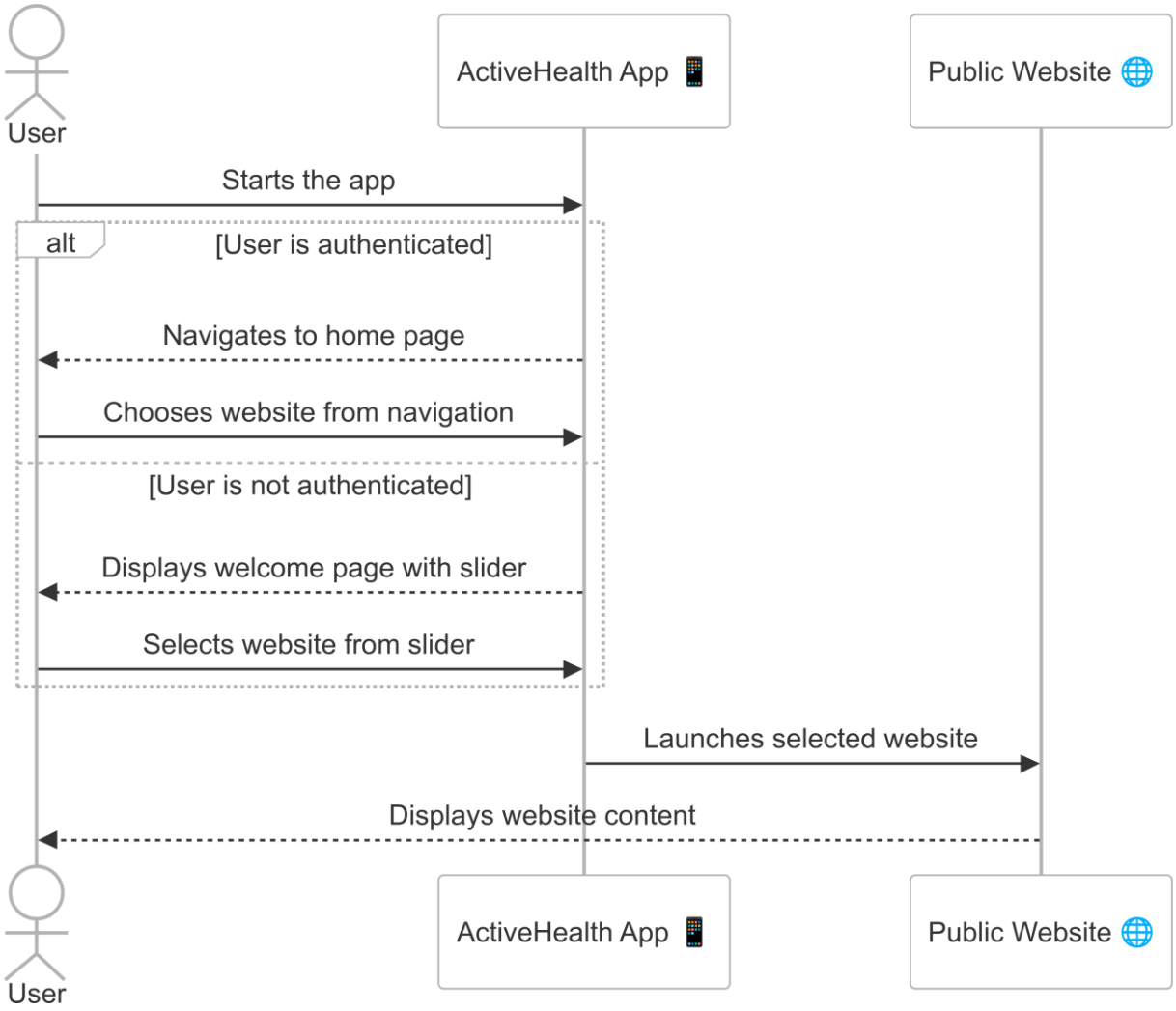

Figure 20: Sequence diagram describing the user interaction with knowledge hub, community network and marketplace

4.12 User Interaction with the Associative Catalog from within the ActiveHealth Application

Table 24 PF12: User Interaction with the Associative Catalog from within the ActiveHealth application.

ID	PF12
Title	User Interaction with the Associative Catalog from within the ActiveHealth application.
Involved Components	ACT, DASH
Actors	Parents
Description	Users want to view the BIO-STREAMS associative catalog of health professionals.
Trigger	User clicks relevant navigation item in app

Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User clicks “Associative Catalog” link
	2	Request is shown to allow access to non-precise geolocation info.
	3	Geolocation is sent to Dashboard API to retrieve relevant professionals in users' location.
	4	Table is displayed in app.
Alternative Flow	**Cause**	**User does not allow access to location info**
	STEP	**DESCRIPTION**
	1	User denies access to geolocation info
	2	A message is shown explaining the problem to the user.
	3	The user may choose to close or go to the Community Network website via external link.

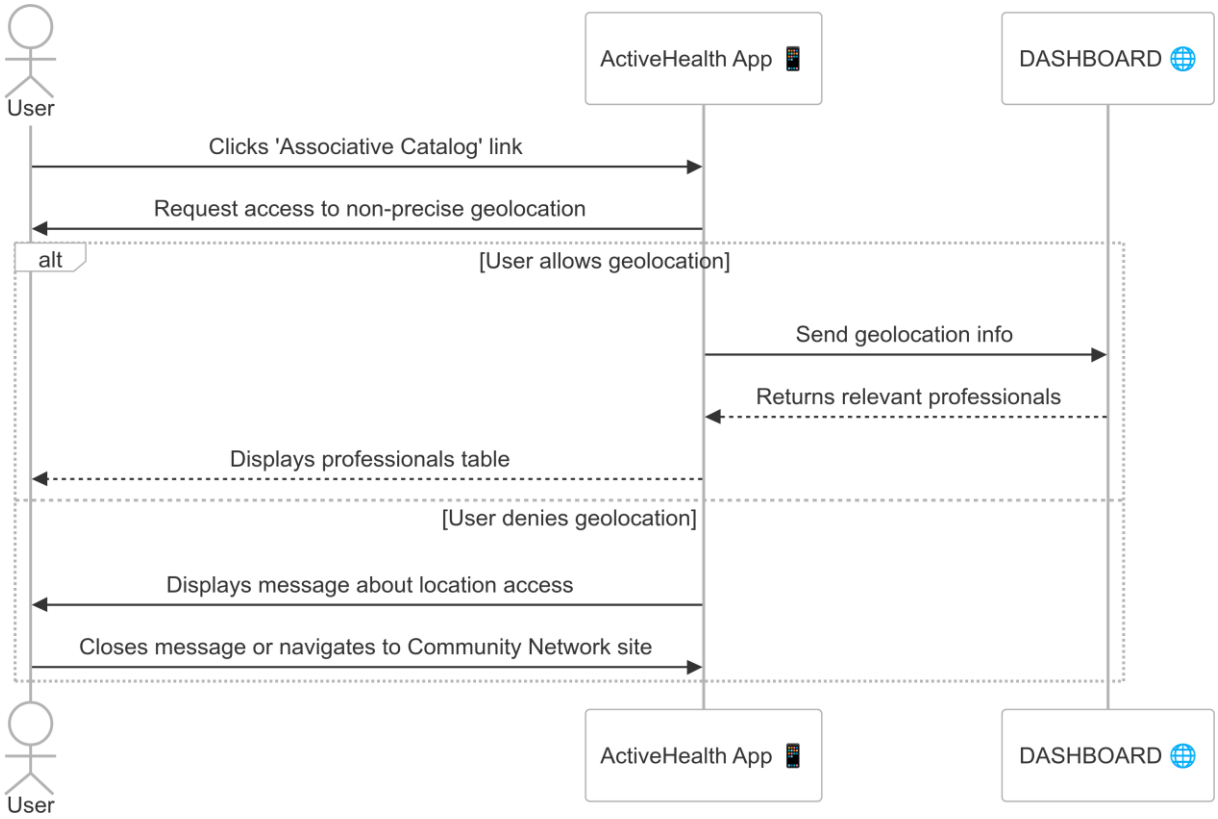

Figure 21: Sequence diagram describing the user interaction with the associative catalog from within the ActiveHealth application.

4.13 User Interaction with Serious Game

Table 25 PF13: User Interaction with Serious Game

ID	PF13	
Title	User Interaction with Serious Game	
Involved Components	DASH, AUTH, SERG, BNB	
Actors	Authorised user (player)	
Description	The user enrolls in the selected game directly from the dashboard by clicking on the Serious Game button. During gameplay, the user is rewarded with achievements or badges and receives real-time feedback based on their actions.	
Trigger	Initiation action in Dashboard	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User visits the Dashboard and performs a successful log-in.
	2	The user launches the Serious Game from the dashboard by clicking the "Start Game" or “Resume Game” button.
	3	During gameplay, the user is rewarded based on their actions. The data collected from these interactions (e.g. screen time play) is temporarily stored on the user's device.
	4	After completing the game, the results are automatically displayed in the application.
	5	The collected data is sent to the BNB through the BNB Gateway Service, where it is stored and available for authorised users.
Alternative Flow	**Cause**	**Failure of User Authentication**
	STEP	**DESCRIPTION**
	1	User cannot log-in because they forgot the password
	2	User clicks “Forgot password” and enters the user id.
	3	Request is made to clinical site and provider resets the password *

	4	User logins again and sets a new password.

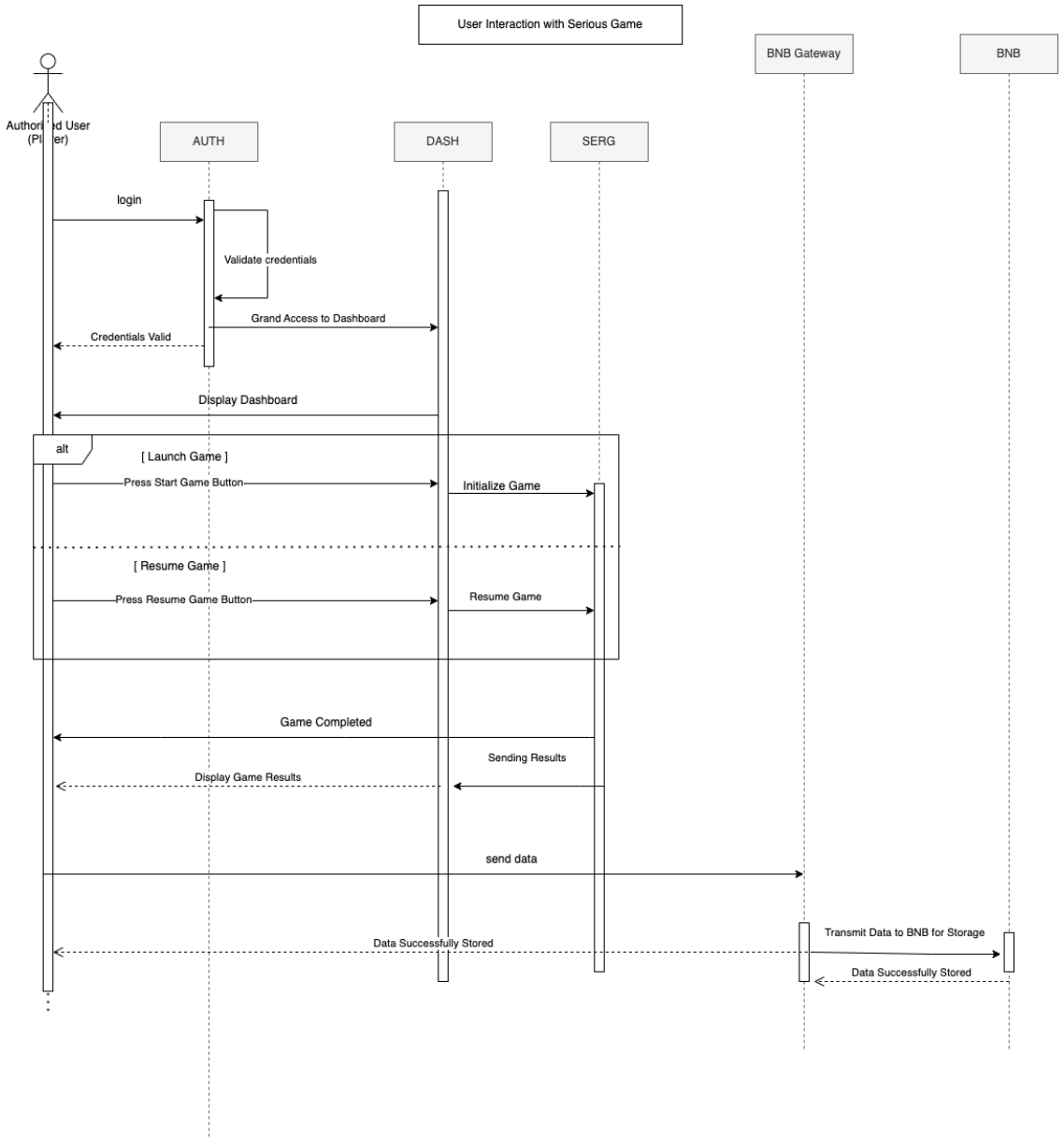

Figure 22: Sequence diagram 1 describing the user interaction with serious game

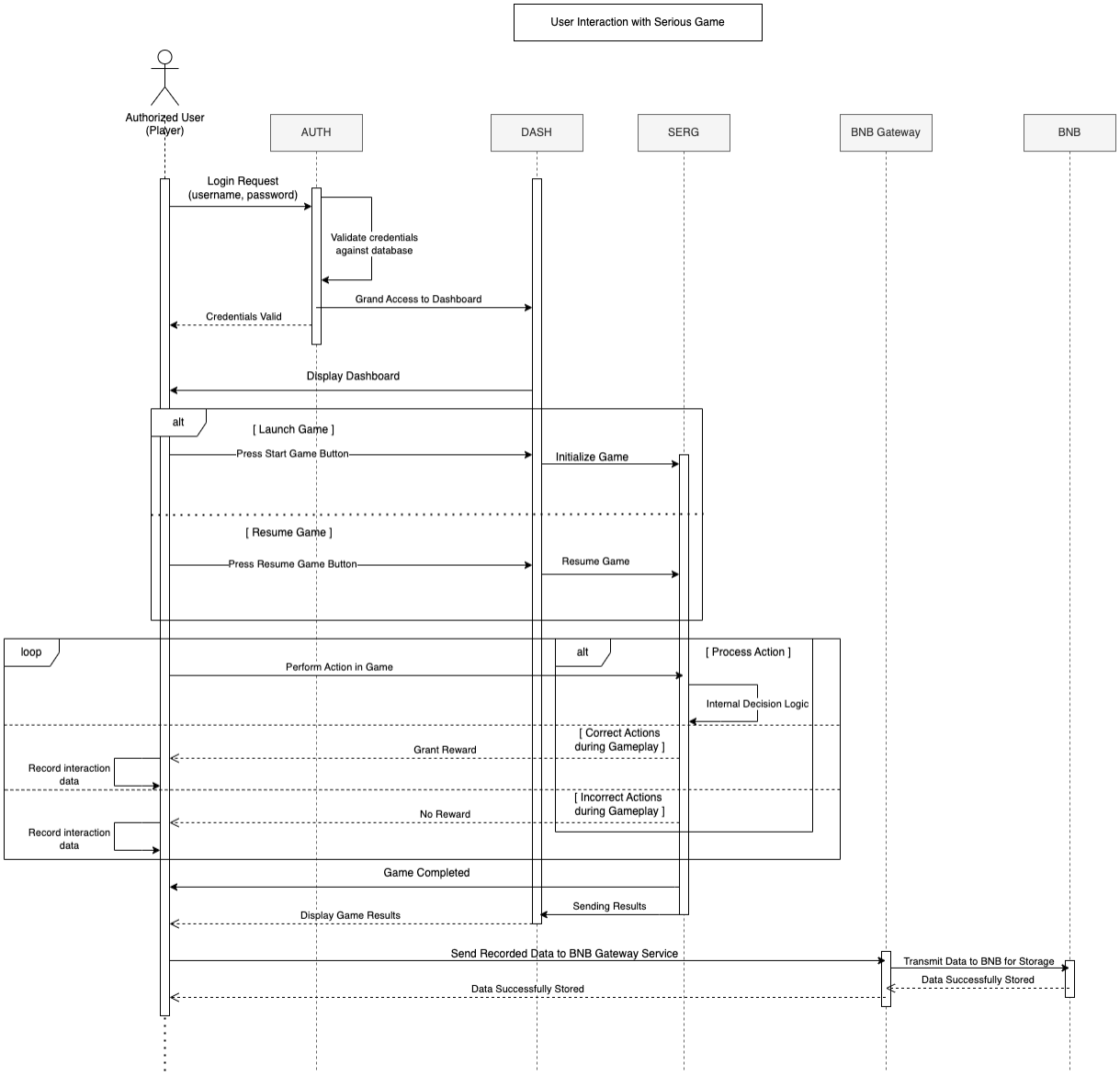

Figure 23: Sequence diagram 2 describing the user interaction with serious game

4.14 Participant in Clinical Studies Opt-out

Table 26 PF14: Participant in clinical studies opt-out

ID	PF14
Title	Participant in clinical studies opt-out
Involved Components	ACT, DASH
Actors	Parents, Doctors
Description	User wants to opt-out from the clinical studies and delete all relevant information

Trigger	User clicks relevant menu item	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	User clicks “Opt-out” from relevant navigation item in the app.
	2	Dialog is shown requesting the users’ real name
	3	User Id, Real name provided, date is stored in the Dashboard
	4	Study administrator receives the request and validates if the real name provided matches the clinical data.
	5	If the name matches, the request is granted, and all data is removed from both the relevant Node Bundle and the dashboard.
	6	Push notification is sent to user that they have been deleted.
Alternative Flow	**Cause**	**Real name provided does not match the clinical data**
	STEP	**DESCRIPTION**
	1	Push notification is sent to the user to contact relevant study authority about their request
	2	The request is examined manually.

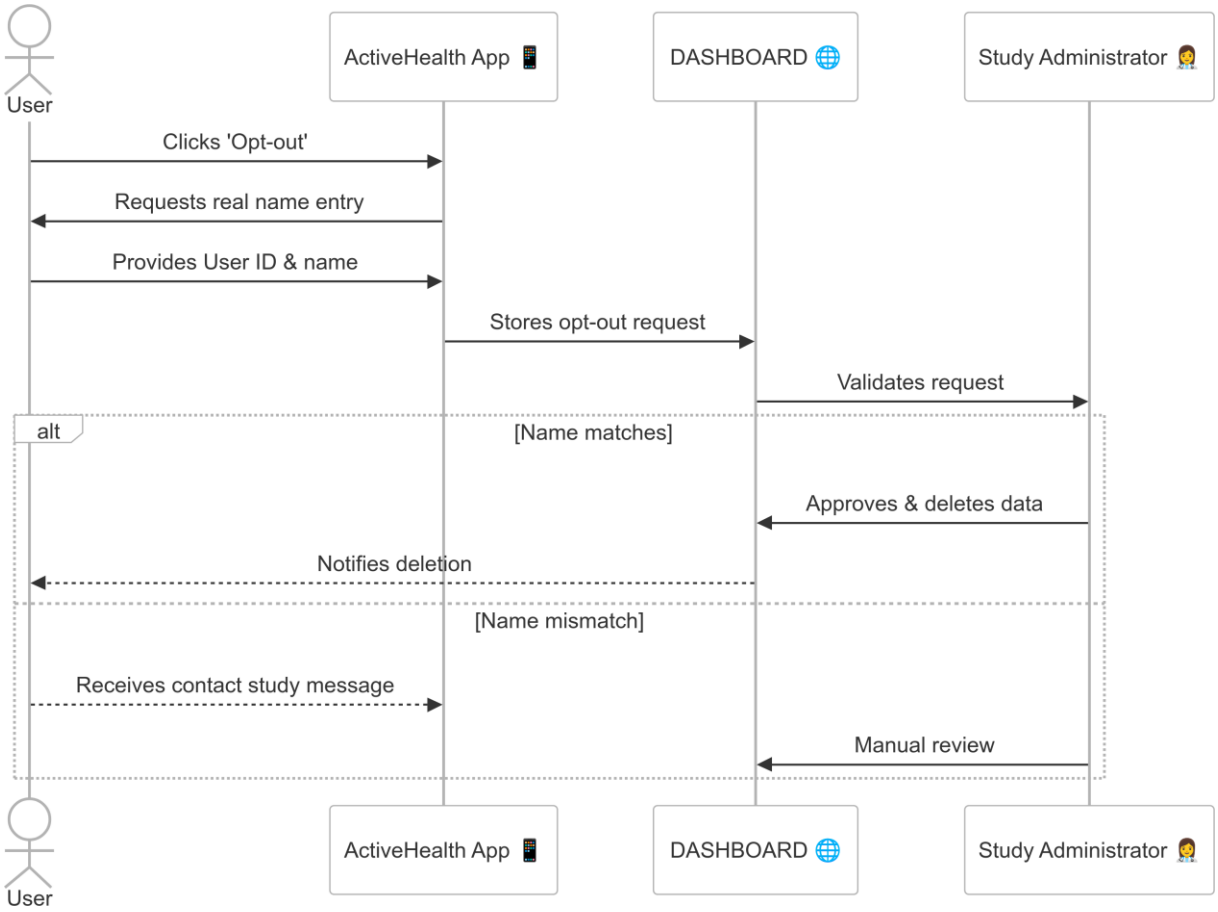

Figure 24: Sequence diagram describing participant in clinical studies opt-out

4.15 Cost Related Data Selection

Table 27 PF15: Cost related data selection

ID	PF15
Title	Cost related data selection
Involved Components	COST, DASH
Actors	Health prevention program manager, healthcare provider
Description	User selects socio-economic datasets to be used in the cost and impact simulation
Trigger	User clicks relevant menu item in the Dashboard

Main Success Scenario	**STEP**	**DESCRIPTION**
	1	The health prevention program manager finds the Cost Model Estimator in the menu item of the BIO-STREAMS Dashboard. (a link redirect to the Cost Model Estimator application where (user sign-up and sign-in to the service)
	2	The user selects the language
	3	the users select the country (BIO-STREAMS pilot countries)
	4	The user selects the socio-economic data source (e.g. Target population; Healthy weight subjects within the target population; Overweight subjects within the target population; Obese subjects within the target population; Mean age of subjects within the target population)
	5	The User can choose the socio-economic data available from the BIO-STREAMS study and already available in the platform dataset
Alternative Flow	**Cause**	**User can prepare other socio-economic datasets for the simulation**
	STEP	**DESCRIPTION**
	1	The user can enter new data from literature
	2	The user can enter new data from the community healthcare system.
	3	The user inserts data autonomously.
	4	The user asks for collaboration to experts in the community (e.g. healthcare providers) to be invited to collaborate in the workflow through a simple mail invitation system

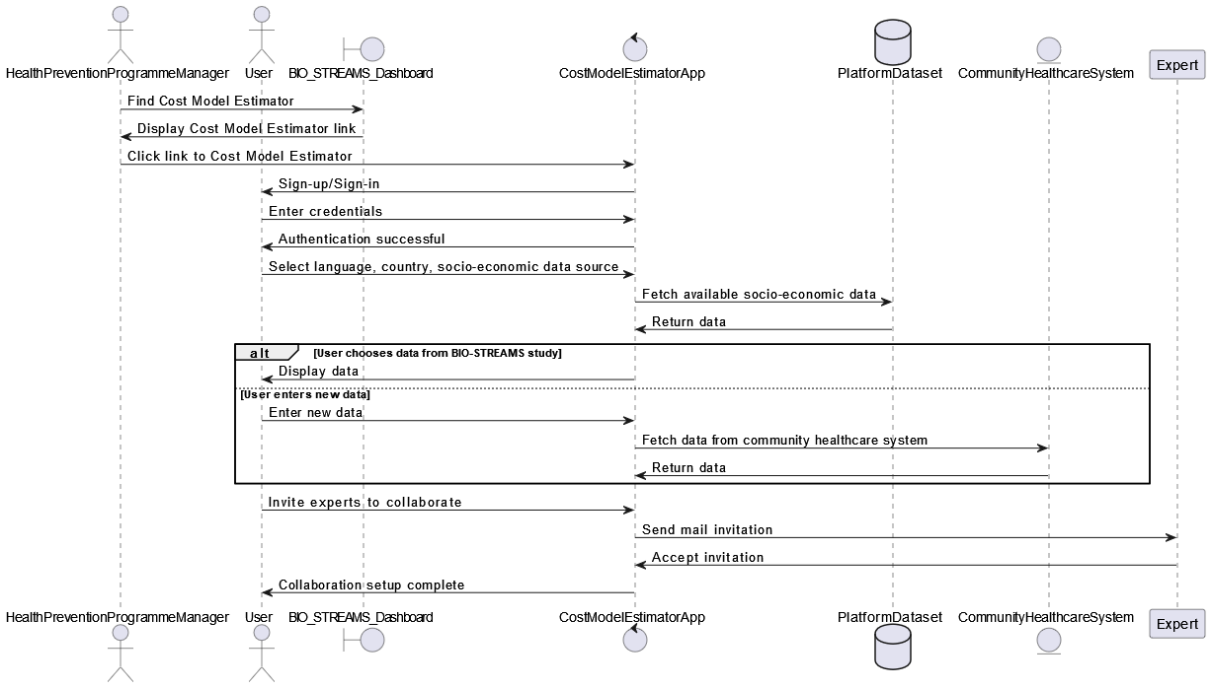

Figure 25: Sequence diagram describing cost related data selection

4.16 Cost-related Short and Long-term Simulation

Table 28 PF16: Cost-related short- and long-term simulation

ID	**PF16**	
Title	Cost-related short- and long-term simulation	
Involved Components	COST	
Actors	Health prevention program manager	
Description	The user defines the type of projection of the cost model e.g. short or long-term	
Trigger	User selects the socio-economic data to be used in the model within the Cost Model Estimator	
Main Success Scenario	**STEP**	**DESCRIPTION**
	1	The health prevention program manager (user) select socio-economic data

	2	The user can select the type of obesity prevention / management program (e.g. nutritional, physical activity, mixed etc.), the source of effectiveness data (e.g. from BIO-STREAMS intervention evidence or from user inputs), the obese subjects percentage variation at 6 months, the obese subjects percentage variation from month 6 to year 2.
	3	The user is to input in the system the estimated or actual cost (if previous data are available) of the intervention program selected.
	4	The user decides what type of projection apply to the cost and impact calculation that, as mentioned in the model description, it can be either « short term » or « long term »
	5	The cost related data model provides the result of the model.

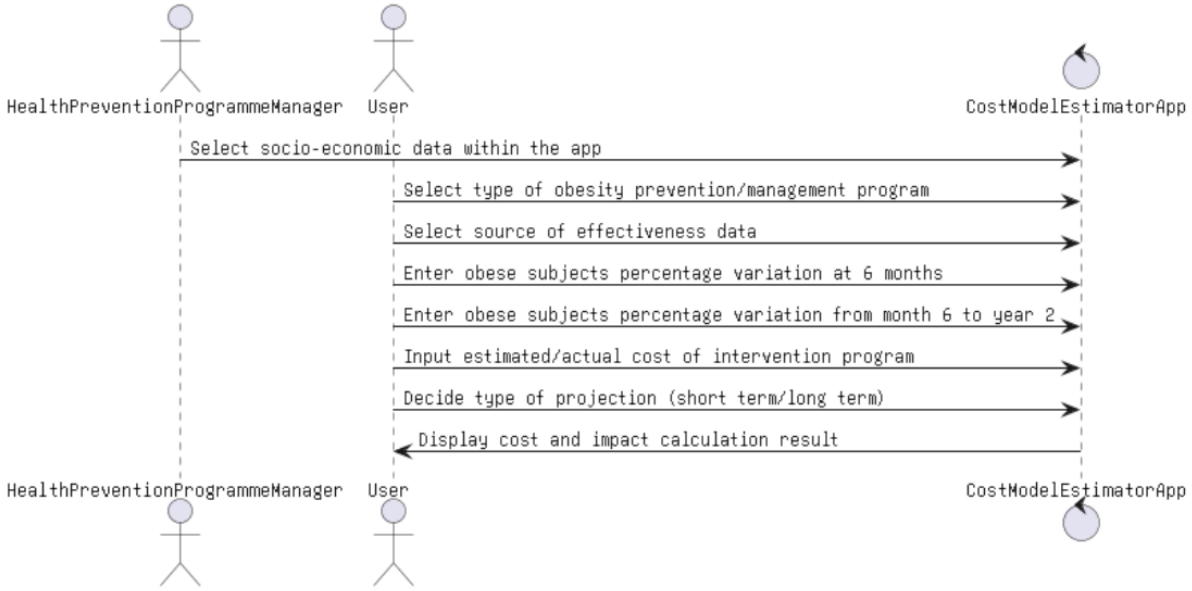

Figure 26: Sequence diagram describing cost-related short- and long-term simulation

5 BIO-STREAMS Platform Deployment (or Physical) View

Server infrastructure is considered the backbone of BIOSTREAMS platform, which includes the hardware such as the servers and networking devices, the software such as the operating systems and the applications. It enables data storage, processing and communication among devices.

5.1 Hardware Specifications – ICCS's Cloud Infrastructure

The key components of the server infrastructure include the hardware, software, networking devices. The hardware comprises all the physical devices that are used in a server infrastructure, which are the servers (high-end hardware and software configuration to store and manage data), the networking equipment (switches, routers and cables) that connect the servers to the network. The software used on the servers is responsible for the management of the Virtual Machines (VMs) management which provides the virtualization technology that assigns virtual versions of the hardware resources. This technology involves hypervisors (software that manages virtual machines on physical servers) which are used to setup virtual machines that are the software-based representations of physical computers running multiple operating systems or applications.

The hardware devices used for hosting the applications of BIOSTREAMS have been setup to support redundancy, which means having duplicates of important data, just in case the original copy doesn't work. It ensures that if something fails, there is always a backup plan. In terms of data redundancy hard disk drives in RAID setup have been setup and backup power supplies are used to automatically take over if the primary power source fails. The infrastructure's ability to handle increasing workloads (scalability) is managed by the administrators of the infrastructure, which ensures that if more resources or hardware upgrades are need, they can be handled. Regular backups of the VMs take snapshots of the data and store copies to secondary storage locations for restoring data in case of loss or corruption. Monitoring tools are utilized to allow the administrators to check server performance and track server metrics (CPU, memory usage, hard disk usage, network traffic, etc.).

For the development of the BIOSTREAMS platform, the VMs that have been setup are running Ubuntu OS and they are used to deploy all the services required for the development and the CI/CD environment. There are currently 3 VMs used for a) continuous development/continuous delivery services, b) development environment, c) production environment.

5.2 ICCS's Cloud Infrastructure Security Practices

Security protocols and guidelines are established to protect servers and data from unauthorized access or threats. They include firewalls, encryption and access control. As Linux is an open source software, it has security flaws must be considered. Securing a Linux server is vital to defend from various threats and to safeguard the data. The following measures have been taken to increase the security of the infrastructure. The measures are frequently revised and if extra safeguards are required, they are deployed in due time.

The most important step in securing the server is updating it often, to take advantage of the latest security revisions. Access to the server is given only to authorized users via privileged

user accounts. Access via SSH is performed only using SSH Key authentication, to safeguard the servers and to provide a more comprehensive protection against various threats. Installing a firewall is one of the most popular and simple ways to protect a server. The Uncomplicated Firewall (UFW) has been used, as a reliable to only accept approved traffic.

5.3 Runtime Deployment Diagram

In this section, we describe the Deployment view (also known as the physical view) in the 4+1 architectural view model, which provides a detailed overview of the system's physical deployment on hardware infrastructure. This view focuses on how the software components are distributed across various physical nodes, including servers, cloud environments, and devices. It helps in understanding the system's performance, scalability, and network communication, detailing the interactions between physical entities such as machines, cloud services, and mobile devices.

The Deployment View ensures that all components are properly placed to meet the system's non-functional requirements like reliability, availability, security, and performance. It plays a crucial role in resource planning and optimizing the infrastructure to support the overall architecture.

In our architecture, we have a combination of cloud-based, on-premises, and mobile deployments. The deployment spans multiple clinical sites, private and public cloud infrastructures, and mobile devices that interact with cloud components. Below is a detailed description of how these components are deployed across different environments:

1. **BNB Components at Clinical Sites:**
    - Each clinical site (from consortium's clinical partners CHUL, BLOCKS, NKUA, UKCM, KI, VHIR, PENTELI) has a dedicated instance of the BNB component, deployed directly on the site's local infrastructure (on-premises). These deployments are crucial for managing local data processing and ensuring compliance with the site's specific regulatory or security requirements.
2. **Citizen BNB in Cloud (other than ICCS's cloud infrastructure):**
    - In addition to the on-site deployments, the Citizen BNB is deployed in a cloud environment. This serves as a centralized instance, aggregating and storing data collected from all mobile apps. The reason the Citizen BNB needs to be in a separate cloud infrastructure compared to the rest of the cloud components is that ICCS's infrastructure is not legally compliant to store such data.
3. **Components Deployed in ICCS's Cloud Infrastructure:**
    - Several other key components of the system—namely IMS, SDG, OPENTK, DASH, NGW, LOGS, METRICS, LOC, COST and SMS—are deployed in a separate cloud infrastructure**,** ICCS's infrastructure. These components are responsible for various processing, data management, and service orchestration tasks within the system. By being hosted in the cloud, they ensure scalability and easy integration with external systems or services.
    - These cloud-based components interact heavily with each other, forming the backbone of the platform, and are critical for processing data received from clinical sites as well as external applications.
4. **Mobile Apps Interacting with Cloud Components:**

- We have four mobile apps (ACT, SERG consisting of SERG1, SERG2, SERG3) that are used for data collection and user interaction. These mobile applications send data to the components deployed in the cloud infrastructures (e.g., IMS, SDG, DASH). The data from the mobile apps might include dietary plan data, behavioral data, or other user inputs, which are then processed and analyzed by the cloud-based components.
- These apps serve as the primary interface for end-users, contributing to a distributed system where mobile devices, cloud, and on-prem infrastructure interact seamlessly.

This deployment setup ensures that sensitive data is handled locally, when necessary (clinical sites), while leveraging the scalability and flexibility of the cloud for broader processing needs. The architecture balances performance, security, and compliance by distributing components strategically across on-premises, private cloud, and public cloud environments, all while maintaining robust communication and data exchange through mobile interfaces.

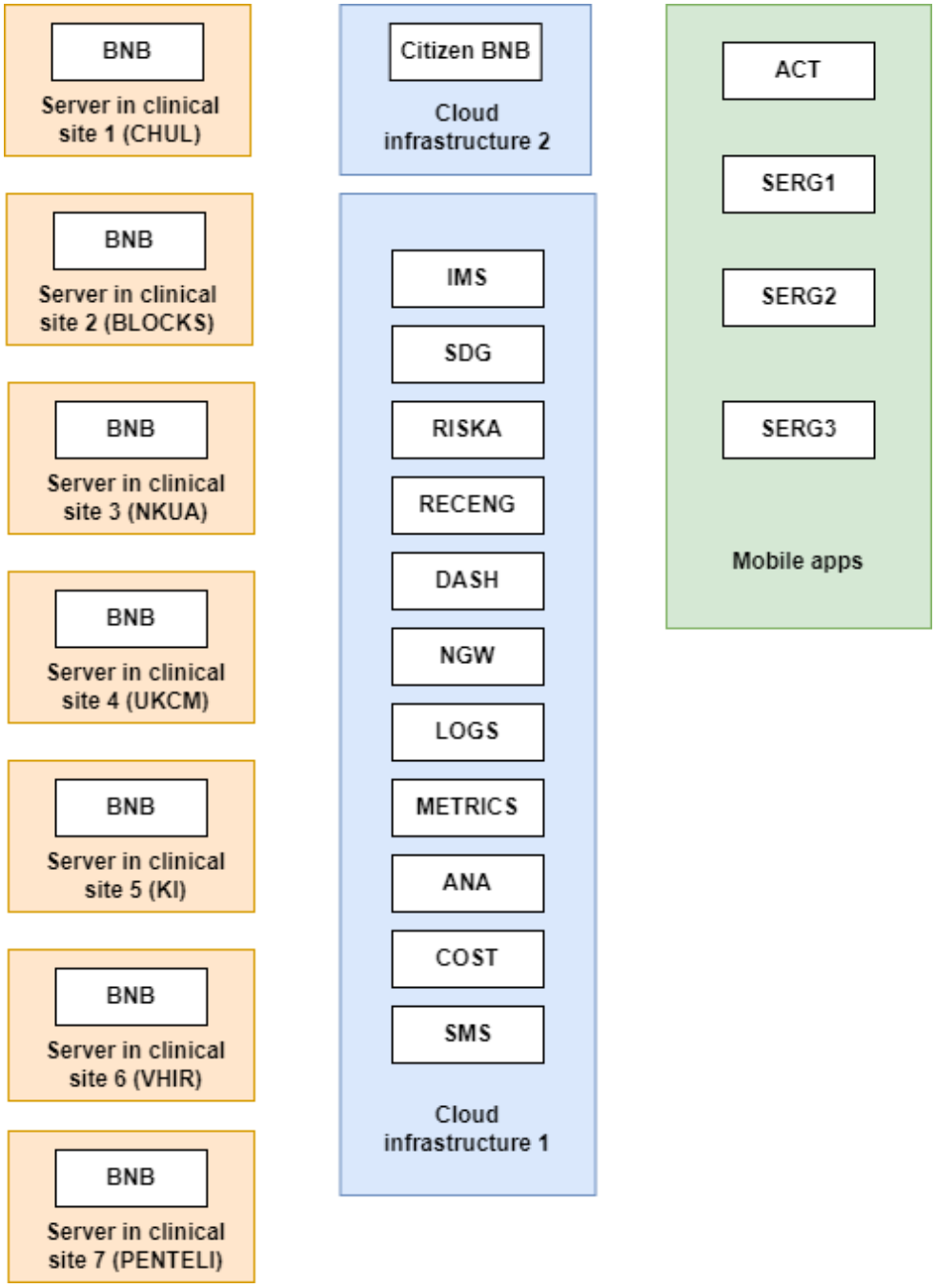

Figure 27: Deployment view of the BIO-STREAMS platform

6 Ecosystem Management Technologies

6.1 DevOps and CI/CD Workflow

DevOps is a methodology that integrates software development (Dev) and IT operations (Ops) with the primary objective of shortening the development lifecycle and delivering high-quality software continuously. A key aspect of DevOps is the implementation of Continuous Integration (CI) and Continuous Delivery (CD) pipelines. CI ensures that developers frequently integrate code into a shared repository, where automated tests and build processes validate changes, catching issues early in the development cycle. Tools like Jenkins orchestrate these pipelines, allowing teams to automate the build, test, and integration processes, minimizing the risk of integration issues and speeding up feedback loops.

Continuous Delivery extends CI by automating the release process, ensuring that software can be safely deployed to production environments at any time. CD pipelines typically include stages for automated testing, security scanning, and performance validation, allowing for frequent, reliable releases. Together, CI/CD enables a seamless path from code development to deployment, ensuring consistency, security, and rapid iteration in modern software delivery practices.

Best DevOps practices and a CI/CD workflow is employed for the BIO-STREAMS development teams to efficiently commit changes, gather and test source code and artifacts as well as deploy the services in the project's servers.

6.2 CI/CD Tooling

In the context of Task 5.6 of the project, we implement DevOps and CI/CD practices using a comprehensive stack of tools designed to support the entire software lifecycle. *GitHub* [12] is used for code management and versioning, enabling efficient collaboration and version control. *Jenkins* [13] handles the automation of building, testing, and deployment, ensuring continuous integration and delivery. *Docker* [14] is used for software packaging, with *Docker Compose* [15] managing multi-container applications. *Harbor* [16] serves as the container repository, while *Portainer* [17] manages Docker deployments and provide insights into registry management. *Keycloak* [18] handles user authentication and role-based access control, ensuring secure access to the CI/CD pipeline. The deployment of the project's platform will eventually span multiple environments (development and production), with robust procedures and practices in place for workflow automation and data management, all underpinned by security measures. This will ensure effective and secure operations across all environments, ensuring the project's success and the release of a fully tested, deployed operational system. More details regarding the CI/CD stack can be found in the subsections below.

6.2.1 GitHub for Code Management

GitHub [12] is a widely used platform for code versioning and collaborative software development, built on top of Git, a distributed version control system. It allows teams to manage the source code of each component in an architecture using repositories, where each repository can maintain its own codebase, documentation, and history of changes. GitHub's branching and pull request mechanisms enable developers to work on isolated features or bug fixes while keeping the main codebase stable. Changes can be reviewed, discussed, and merged via pull requests, ensuring quality and transparency across the team. BIO-STREAMS GitHub organization can be found at https://github.com/orgs/BIO-STREAMS-eu-project.

In BIO-STREAM's CI/CD architecture, each component's repository will also contain its own pipeline configuration (Jenkinsfile), defining the build, test, and deployment processes. These pipelines are triggered automatically on specific events, such as code pushes or pull requests, ensuring that the latest changes are continuously tested and integrated. GitHub's permission system allows adding contributors, who can collaborate by submitting code, raising issues, and reviewing changes, all while maintaining control over what gets merged into the codebase. This promotes collaboration across multiple teams and contributors in a secure, auditable way.

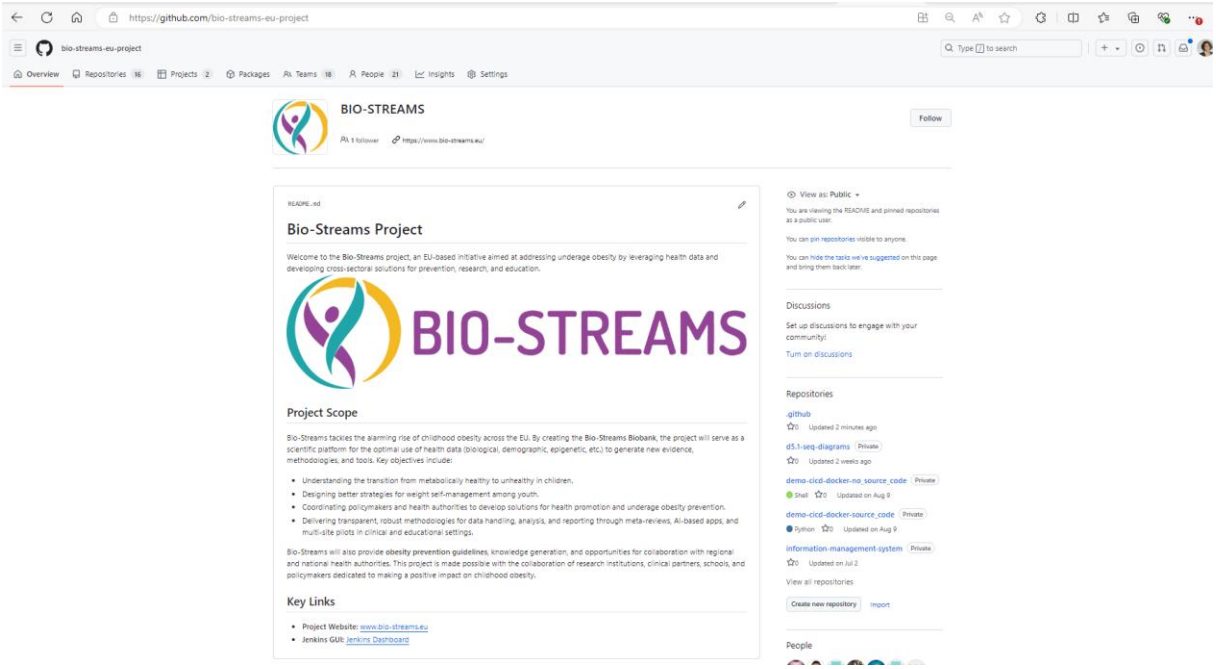

Figure 28: BIO-STREAMS GitHub organization

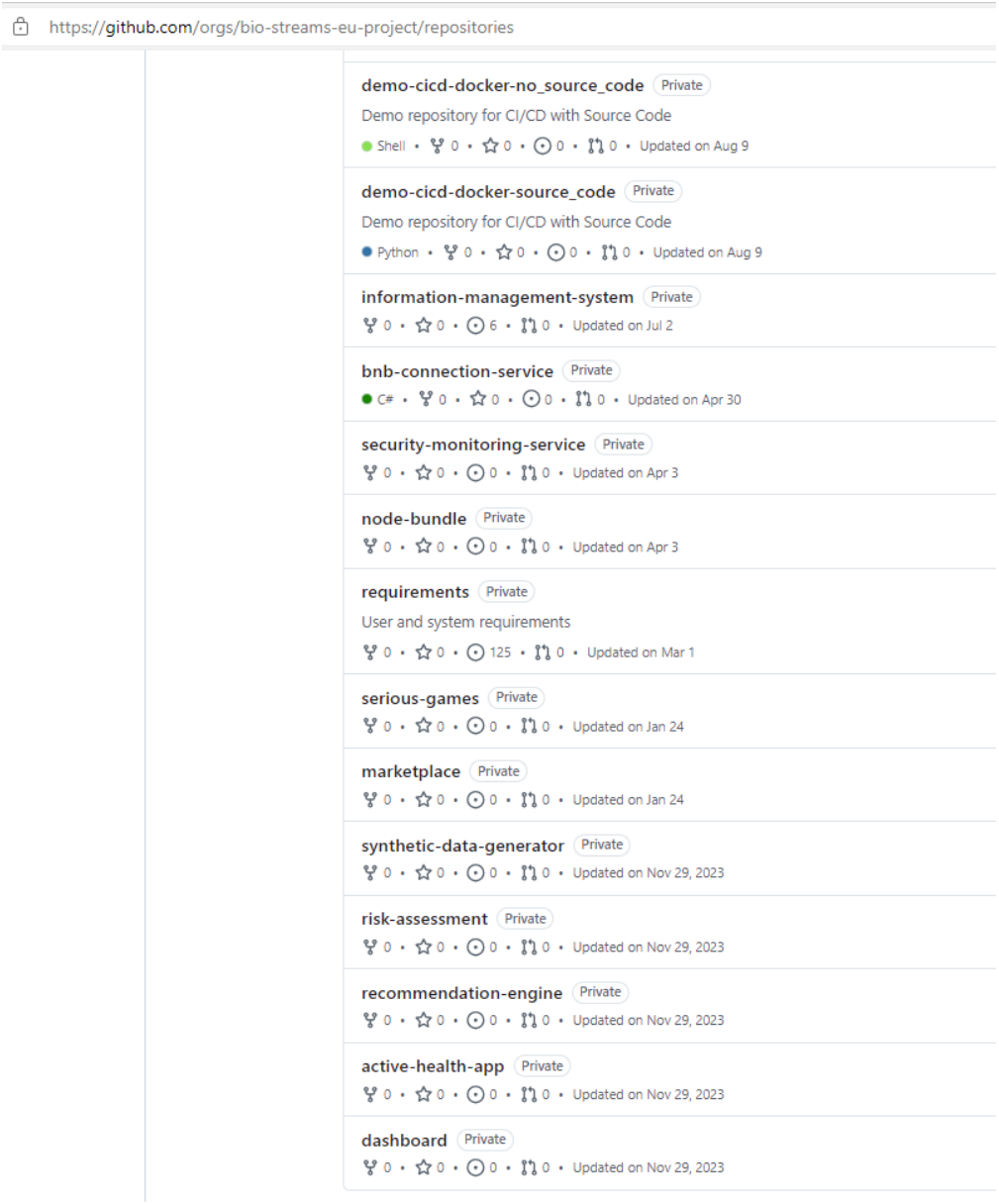

Figure 29: Listing of BIO-STREAMS GitHub repositories

6.2.2 Jenkins for CI/CD automation

Jenkins [13] is a widely used open-source automation server designed to facilitate the continuous integration (CI) and continuous deployment (CD) of software projects. It streamlines the process of building, testing, and deploying applications, allowing teams to automate repetitive tasks and ensure that code changes are integrated smoothly into production environments. Jenkins is highly extensible, offering a wide range of plugins to integrate with other tools and systems commonly used in CI/CD pipelines. It supports pipeline-based deployments, enabling teams to define workflows as code (e.g., using Jenkinsfiles), making the automation process transparent, repeatable, and easily manageable.

Jenkins web interface is available at https://jenkins.BIO-STREAMS.eu.

In a typical development and deployment workflow, Jenkins is triggered by webhooks that respond to code changes or image updates. For example, when a developer pushes code to a GitHub repository, a GitHub webhook triggers Jenkins to start a build pipeline. The pipeline will build the application, run unit tests, and package the software into a Docker container

image. Once the build passes, Jenkins can push the Docker image to a container registry such as Harbor. Additionally, Jenkins can be triggered by a webhook from Harbor when new container images are uploaded or updated, initiating the deployment process. Jenkins can then pull the updated image from Harbor and deploy it to the designated environment (Docker).

To ensure security and manage access to Jenkins pipelines, a role-based access control (RBAC) system is integrated with Keycloak. This limits access to specific Jenkins folders based on user roles, allowing only authorized members of each component owner's technical team to trigger or view pipelines. This ensures that only authorized personnel can manage deployments and access sensitive operations.

Here's an example of a Jenkinsfile that logs into Harbor using credentials stored in environment variables (HARBOR_USER and HARBOR_PASSWORD), pulls the latest version of the application image from Harbor, stops and removes any existing container running the old version of the application, and runs the new Docker container with the updated image:

```
pipeline {
    agent any

    environment {
        HARBOR_URL = 'harbor.BIO-STREAMS.eu'
        HARBOR_PROJECT = 'BIO-STREAMS'
        HARBOR_IMAGE = 'BIO-STREAMS-app'
        HARBOR_TAG = 'latest'
        DOCKER_CONTAINER_NAME = 'BIO-STREAMS-app-container'
    }

    stages {
        stage('Login to Harbor') {
            steps {
                script {
                    sh "docker login -u ${HARBOR_USER} -p ${HARBOR_PASSWORD} ${HARBOR_URL}"
                }
            }
        }

        stage('Pull Image from Harbor') {
            steps {
                script {
                    sh "docker pull
${HARBOR_URL}/${HARBOR_PROJECT}/${HARBOR_IMAGE}:${HARBOR_TAG}"
                }
            }
        }

        stage('Stop and Remove Existing Container') {
            steps {
                script {
                    sh """
                    if [ \$(docker ps -q -f name=${DOCKER_CONTAINER_NAME}) ]; then
                        docker stop ${DOCKER_CONTAINER_NAME}
                        docker rm ${DOCKER_CONTAINER_NAME}
                    fi
                    """
                }
            }
        }

        stage('Run New Docker Container') {
            steps {
                script {
                    sh "docker run -d --name ${DOCKER_CONTAINER_NAME}
${HARBOR_URL}/${HARBOR_PROJECT}/${HARBOR_IMAGE}:${HARBOR_TAG}"
                }
            }
        }
    }
```

```
    post {
        always {
            script {
                sh "docker logout ${HARBOR_URL}"
            }
        }
    }
}
```

The integration of Jenkins with both GitHub and Harbor via webhooks ensures that builds and deployments are automated, efficient, and secure, while RBAC integrated with Keycloak provides fine-grained control over access to Jenkins pipelines, safeguarding the deployment process.

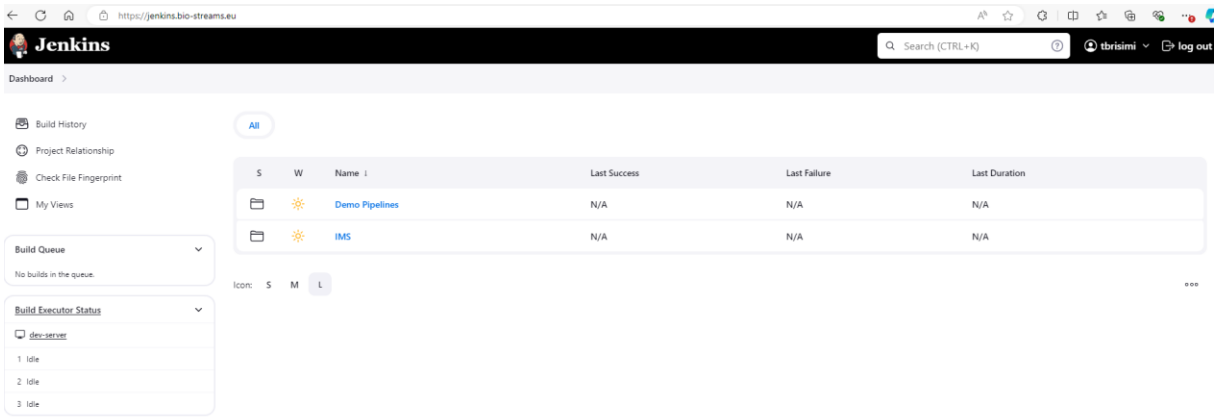

Figure 30: Jenkins folders: one for the demo pipelines and one for the IMS. More folders will be created for all BIO-STREAMS components

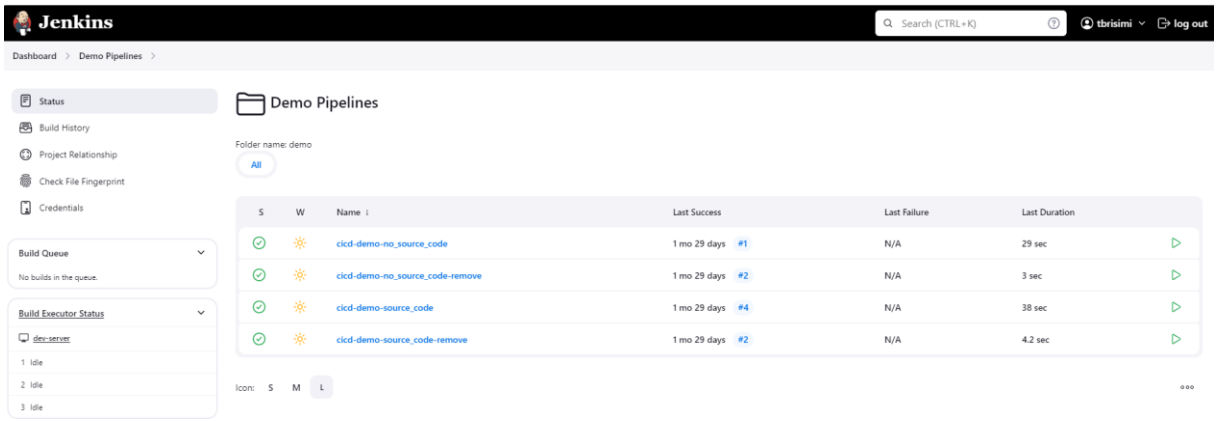

Figure 31: Demo pipelines deployed successfully

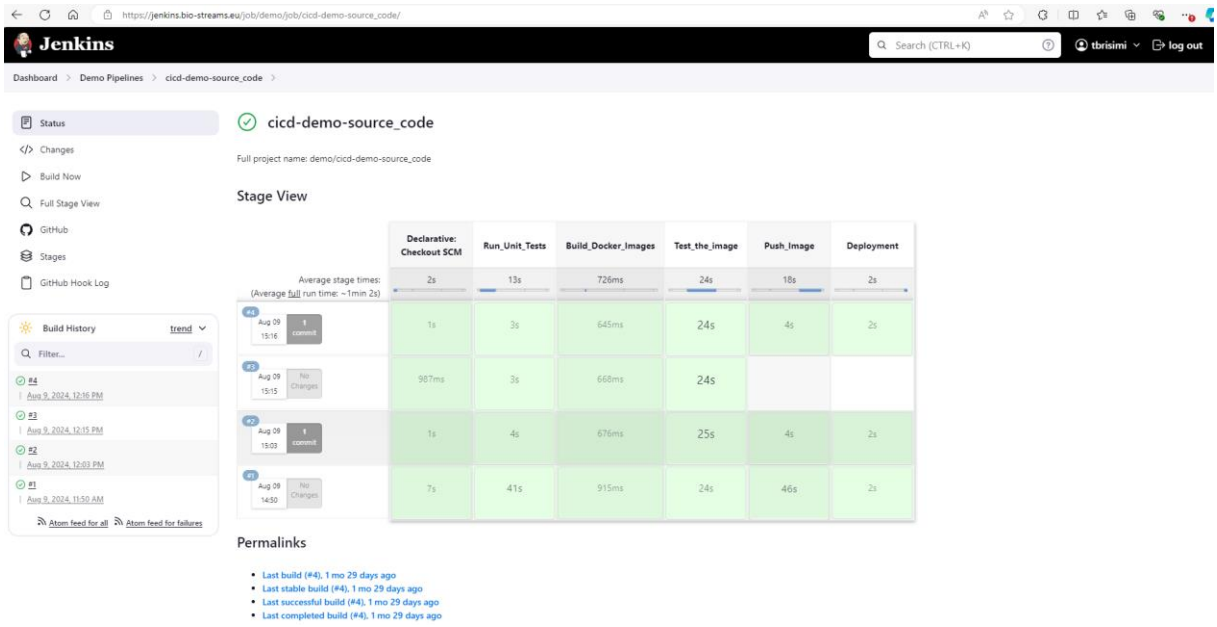

Figure 32: Demo pipelines deployed successfully

6.2.3 Docker for Software Packing and Docker Compose for Docker Application Management

Docker [14] provides an efficient solution for packaging software by encapsulating an application and all its dependencies, configurations, and libraries into a Docker image. These images are portable, lightweight, and ensure consistency across environments, from development to production. With Docker, teams can build once and run anywhere, eliminating compatibility issues that typically arise when moving software between different environments. In a CI/CD pipeline, Docker ensures that the application behaves the same way regardless of the system it's deployed on, thus streamlining testing, deployment, and scaling. Once the Docker image is built, it can be stored in a registry (Harbor in our case) and deployed across various environments seamlessly.

Docker Compose [15] extends this functionality by simplifying the management of multi-container applications. With Docker Compose, developers can define multiple services, networks, and volumes in a single YAML file (docker-compose.yml). This file allows defining how various containers interact, enabling easy orchestration of multi-service applications such as microservices or complex architectures that require multiple components like databases, caches, and backend services. Docker Compose makes it easier to run and test entire applications locally, and in a CI/CD environment, it helps automate the deployment of multiple services together, ensuring that all components of an application are correctly deployed, configured, and started in the proper order.

6.2.4 Harbor as Container Repository

Harbor [16] is an open-source, cloud-native container registry that provides advanced features for managing Docker images and container repositories with an emphasis on security, compliance, and performance.

The web interface of Harbor can be found at https://harbor.BIO-STREAMS.eu.

Harbor integrates with Keycloak and one of Harbor's key features is its role-based access control (RBAC), which allows administrators to define user permissions and ensure that only authorized users can access specific container images.

6.2.5 Portainer for Registry Management

Portainer [17] is a container management platform designed to simplify the deployment, management, and monitoring of Docker environments. It provides a user-friendly web interface that allows developers and operations teams to manage their Docker containers, images, volumes, and networks across multiple environments without needing to use complex CLI commands.

For monitoring Docker deployments, Portainer offers real-time insights into running containers, resource usage (CPU, memory, storage), and the overall health of the environment. It allows users to easily deploy, stop, restart, or scale containers, as well as manage services, networks, and volumes through its intuitive dashboard. Additionally, Portainer supports role-based access control (RBAC), enabling teams to manage permissions for different users and improve security across the deployment process. In a CI/CD pipeline, Portainer can be used to streamline operational tasks, monitor deployed services, and manage containerized workloads effectively.

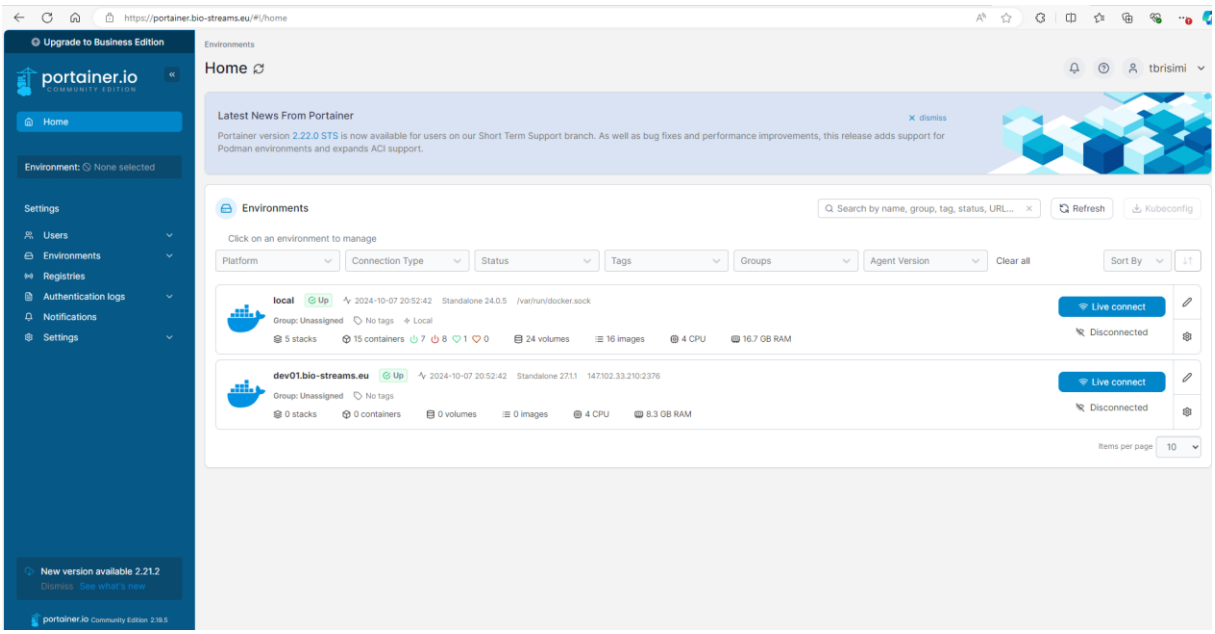

Figure 33: Portainer web interface

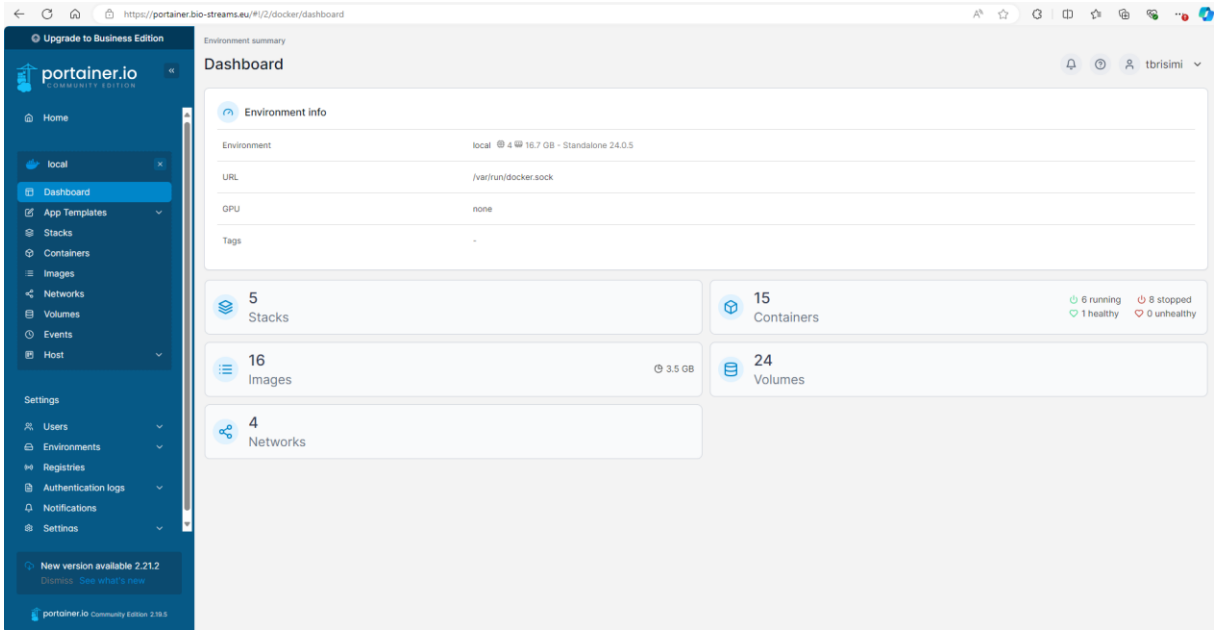

Figure 34: Portainer Dashboard

6.2.6 Keycloak for Access Control

Keycloak [18] is an open-source identity and access management (IAM) solution that provides centralized authentication, authorization, and user management for applications, including CI/CD platforms. In the context of CI/CD, where multiple users, teams, and contributors are involved, Keycloak streamlines user authentication and role-based access control (RBAC). It integrates with various CI/CD tools (e.g., Jenkins, Portainer, Harbor) to ensure that access is securely managed across the entire pipeline.

Keycloak's administrator's interface can be found at https://keycloak.BIO-STREAMS.eu.

Keycloak supports Single Sign-On (SSO), allowing users to authenticate once and gain access to all CI/CD tools without needing to log in repeatedly. Through Keycloak's administrative console, teams can manage users, define groups and roles, and set up fine-grained permissions to control who can view or modify CI/CD pipelines, deployments, and infrastructure. By integrating Keycloak into the CI/CD solution, we can ensure secure, streamlined access management, and meet compliance requirements for user authentication and auditing.

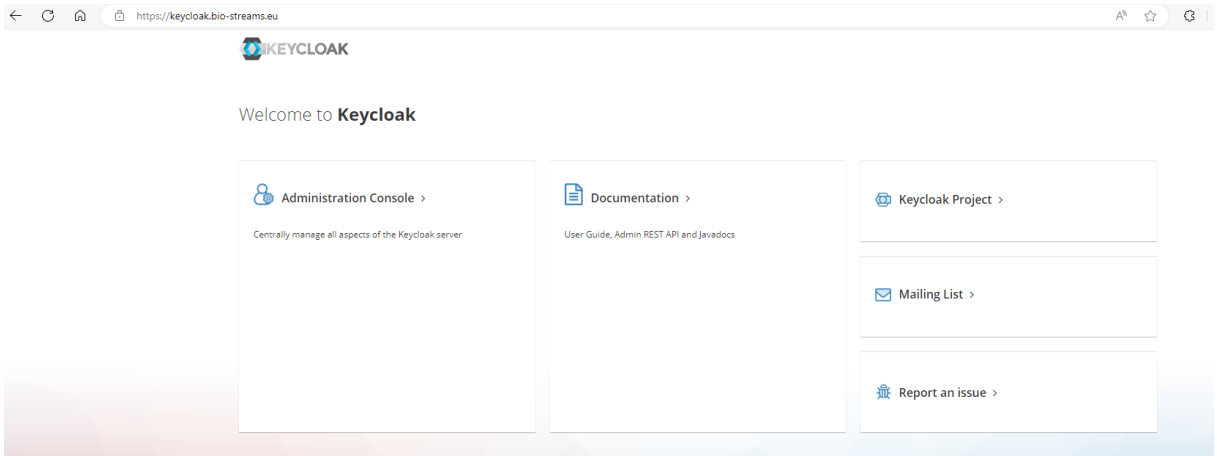

Figure 35: Keycloak admin interface

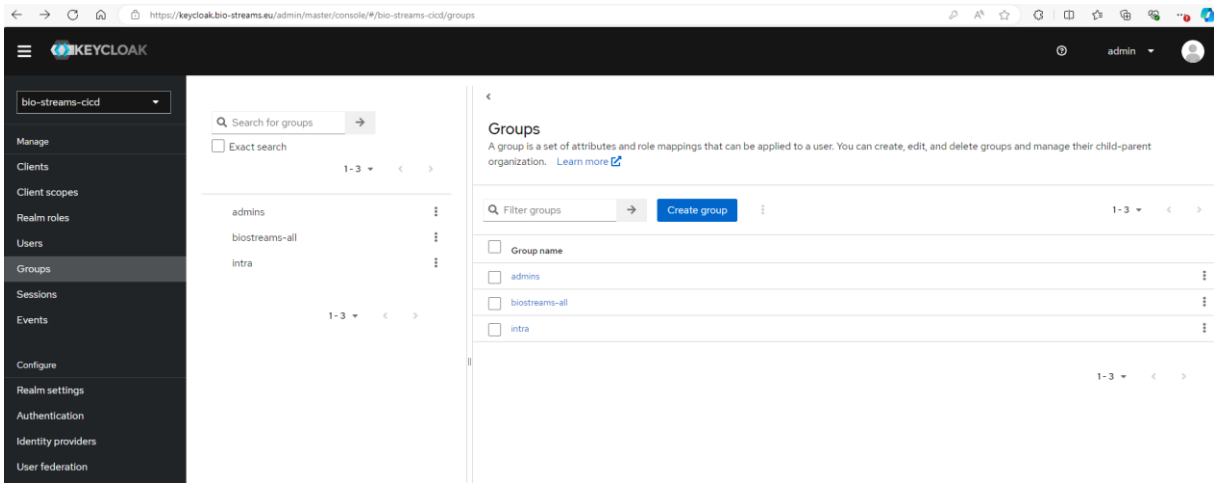

Figure 36: Defined groups in Keycloak

6.3 Continuous Localisation with Weblate

The digital interventions that will be evaluated in the clinical studies include populations from 9 different countries. The following languages have been identified based on the requirements of Study 2 and Study 3.

Table 29: Languages used in BIO-STREAMS pilots

Language Code	Language Name	Pilots
en	*English*	*Base language*
el	Greek	National and Kapodistrian University of Athens (NKUA), Penteli General Children's Hospital (PENTELI)
es	Spanish	Vall d'Hebron Research Institute (VHIR)
fr	French	University Hospital of Liège (CHUL)
sv	Swedish	Karolinska Institute (KI)
bg	Bulgarian	Blocks Health and Social Care EOOD (BLOCKS)
sl	Slovenian	University Clinical Centre Maribor (UKCM)
nl	Dutch	School intervention
pt	Portuguese	School intervention
da	Danish	School intervention

The digital tools that will require localization are the ActiveHealth application and messages (push notifications) sent by the Backoffice of the ActiveHealth app. If required, the Dashboard and Serious Games can also be localized using the methods described below.

To coordinate such a large amount of translation work with 10 different partners, especially in an agile methodology used in the app's development (frequent changes) only a web-based

localization tool can be used. Translation is required not only in the string resources within the apps, but also for push notifications that the users will receive that will enhance the micro-moments aspect of the interventions.

The open-source solution *Weblate* [19] will manage translations, ensuring that text is localized for the different languages required. It will also enable efficient updates to translations as new features and content are introduced, maintaining consistency across all localized versions.

The following workflow will be used for localization:

- Applications are developed and push notification messages are written in the base language common in the project, English.
- Internally, all strings are marked in the source code using a specific GNU gettext [20] based format.
- A special tool extracts all strings from the source code, and updates a *.pot* file that contains all the strings plus the context they are being used.
- Updated .pot files are committed to a specific repository in the BIO-STREAMS GitHub account.

At this stage the Localization Service takes over and

- The tool reads automatically the updated .pot files from the GitHub repository.
- Notifies via e-mail the respective partners that updates need to be made.
- Provides a user-friendly web-based environment for less technically inclined partners to be able to edit the translations.
- Automatically commits changes to the git repository, to provide versioning of the work done.

For the last step, the updated *.po* files for each language are downloaded from the git repository and used in the updated version of the application.

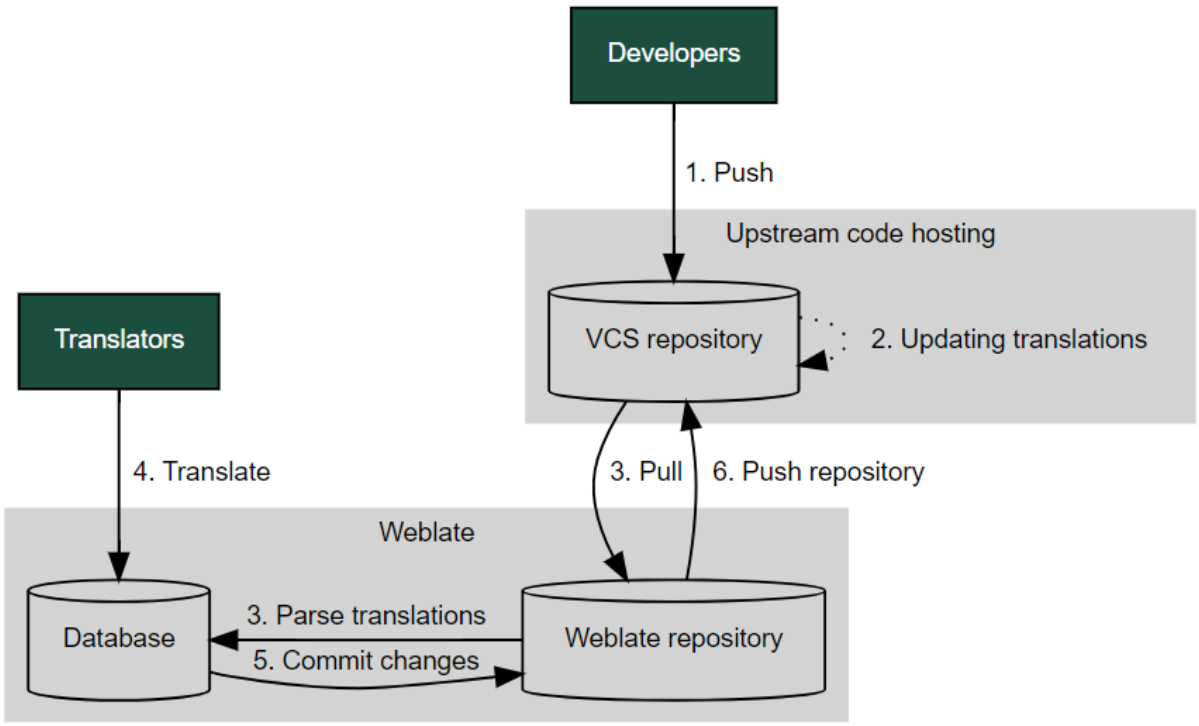

Figure 37: Continuous localization workflow within BIO-STREAMS

7 BIO-STREAMS Platform Integration

7.1 Methodology

To ensure a smooth and systematic approach to integrating various components of the BIO-STREAMS platform, we will follow a structured integration methodology across three release stages: Alpha, Beta, and Final. This methodology focuses on building robust data flows and ensuring seamless communication between all systems, allowing for iterative improvements and feedback at each stage. The following outlines the key steps of our integration process:

1. **Understanding System Architecture & Clarifying Data Flows**

In the initial phase, we thoroughly analyze the existing system architecture and clarify the data flows and interactions between all components. This step ensures that all dependencies and communication requirements are fully understood and mapped out, creating a foundation for integration. Understanding these data exchanges early is crucial for identifying how the components will interact during the Alpha, Beta, and Final releases.

2. **Definition of Integration Points & Communication Protocols**

Following the architectural analysis, we will define the key integration points, which are the specific locations where systems will connect and exchange data. These points are documented in an Integration Matrix, detailing:

- Senders and receivers involved in data exchanges.
- Communication protocols (e.g., REST APIs, message queues, etc.) that will facilitate these interactions.

This matrix is presented in the section below, and might evolve through each release, refining as we move from Alpha to Final, ensuring comprehensive coverage of all integration needs.

3. **Development of Integration Components**

After defining the integration points, the next step involves the development of the necessary integration components. These include APIs, connectors, and adapters that will handle data transformation, and secure communication between systems. Integration components will be iteratively developed and updated in each release cycle, starting with the core functionality in the Alpha stage and expanding to more complex interactions in Beta and Final releases.

4. **Testing & Validation**

Thorough testing and validation will be conducted throughout the integration process to ensure that the system behaves as expected. Testing will be carried out in multiple stages:

- *Unit Testing*: Verifying the functionality of individual integration components in isolation.
- *Integration Testing*: Ensuring that communication between different systems is correctly implemented and that data is exchanged as expected.
- *End-to-End Testing*: Validating complete workflows across all components, testing the entire system as a cohesive unit.

5. **Deployment in Releases**

The integration will be rolled out in three phases:

- *Alpha Release*: A preliminary release focusing on core integration points and fundamental data exchanges. This release will establish the basic infrastructure and set the stage for more complex integrations.
- *Beta Release*: An enhanced release that incorporates more intricate workflows and additional integration points. During this phase, most components will be integrated, and any gaps or issues identified in Alpha will be addressed.
- *Final Release*: The fully integrated system, incorporating all components and ensuring stability, scalability, and full functionality. This release will be the basis for full-scale deployment and operational use.

6. **Documentation**

Throughout the integration process, comprehensive *documentation* will be maintained. This will include technical details on integration points, communication protocols, data flows, and troubleshooting guides. As the integration matures across each release, the documentation will be updated to reflect changes, serving as a key resource for developers, testers, and operations teams.

7. **Monitoring & Feedback Loops for Future Releases**

After each release, monitoring will be conducted to track system performance, data flow efficiency, and potential errors. A feedback loop will be established to capture insights from testing, stakeholders, and end-users. This feedback will be analyzed and used to refine integration components and processes for the next release, ensuring continuous improvement and adaptability throughout the lifecycle of the project.

By following this integration methodology, we ensure that each release builds progressively on the previous one, ultimately leading to a fully integrated and optimized platform in the Final release. This iterative approach allows for flexibility, thorough testing, and alignment with project objectives, while minimizing risk and ensuring smooth transitions between each phase.

7.2 Integration Matrix

The table below, which outlines the communication interfaces, was created by extracting data from the integration matrix shared by INTRA with the technical partners. It provides a comprehensive overview of the complex network of connections and interactions between various platform components. This table identifies the data exchange initiators, the receivers, and the type of communication and security protocols facilitating these interactions. Each row represents a specific integration point, showcasing the relationships between components and defining the communication initiations, which are crucial for seamless integration across the project's infrastructure. These interaction points were identified to support development activities and enable bilateral communication between systems. These interfaces will be pivotal for building complex operational workflows during future integration testing. It is essential to thoroughly test and operationalize these interactions to facilitate the validation of end-to-end trial scenarios involving multiple interfaces. The final version of the communication interfaces will be completed after the development of the components has been completed and insights from testing activities have been incorporated.

Table 30: Integration table between BIO-STREAMS platform components

ID	Component A	Direction (initiator)	Component B	Type of communication	Security
BNB-NGW	Node Bundle	A→B A ←B	Node Gateway	SSH	SSH
SMS-BNB	Security Monitoring Service	A→B	Node Bundle	Log shipping	TLS
NGW-AUTH	Node Gateway	A→B	Auth Server	REST API (request/response)	TLS
NGW-METRICS	Node Gateway	A→B	Metrics Server	REST API (request/response)	TLS
NGW-LOGS	Node Gateway	A→B	Distributed Logs Server	REST API (request/response)	TLS
IMS-NGW	Information Management System	A→B	Node Gateway	SSH	SSH
IMS-BNB	Information Management System	A→B	Node Bundle	REST API (request/response) Via the Node Gateway	TLS
IMS-AUTH	Information Management System	A→B	Auth Server	REST API (request/response)	TLS
IMS-SDG	Information Management System	A→B	Synthetic Data Generator	REST API (request/response)	TLS
SMS-IMS	Security Monitoring Service	A→B	Information Management System	Log shipping	TLS
DASH-RECENG	Dashboard	A→B	Recommendation Engine	REST API (request/response)	TLS

AHA-DASH	Active Health App	A→B A←B	Dashboard	Web sockets	TLS
AHA-AUTH	Active Health App	A→B	Auth Server	REST API (request/response)	TLS
AHA-SMS	Security Monitoring Service	A→B	Active Health App	Log shipping	TLS
AHA-NGW	Active Health App	A→B	Node Gateway	REST API (request/response)	TLS
AHA-LOGS	Active Health App	A→B	Distributed Log Server	REST API (request/response)	TLS
AHA-METRICS	Active Health App	A→B	Metrics Server	REST API (request/response)	TLS
AHA-ANA	Active Health App	A→B	Analytics Server	REST API (request/response)	TLS
DASH-IMS	Dashboard	A→B	Information Management System	REST API (request/response)	TLS
DASH-RISKA	Dashboard	A→B	Risk Assessment Tool	REST API (request/response	TLS
DASH-AUTH	Dashboard	A→B	Auth Server	REST API (request/response)	TLS
DASH-NGW	Dashboard	A→B	Node Gateway	REST API (request/response)	TLS
DASH-METRICS	Dashboard	A→B	Metrics Server	REST API (request/response)	TLS
DASH-LOGS	Dashboard	A→B	Distributed Logs Server	REST API (request/response)	TLS
COST-NGW	Cost Model Estimator	A →B	Node Gateway	REST API (request/response)	TLS
SERG-NGW	Serious game	A→B	Node Gateway	REST API (request/response)	TLS

SERG-ANA	Serious game	A→B	Analytics Server	REST API (request/response)	TLS
SERG-AUTH	Serious game	A→B	Auth Server	REST API (request/response)	TLS
ANA-AUTH	Analytics Server	A→B	Auth Server	REST API (request/response)	TLS
METRICS-AUTH	Metrics Server	A→B	Auth Server	REST API (request/response)	TLS
LOGS-AUTH	Logs Server	A→B	Auth Server	REST API (request/response)	TLS
SMS-AUTH	Security Monitoring Service	A→B	Auth Server	REST API (request/response)	TLS

7.3 Integration Execution Time Plan

The table below offers an overview of all the components identified for the BIO-STREAMS Platform, organized by tasks and partners. Each row corresponds to a specific component, with columns detailing the component name, associated partner, and involved functionalities contained in each of three releases. The *Alpha release*, scheduled for Month 24 (M24), is an internally agreed-upon version developed to meet the specific needs of the project's pilots. The *Beta release*, due in Month 30 (M30), contributes directly to the deliverable D5.3, which is expected by Month 36 (M36). Finally, the *Final release*, planned for Month 36 (M36), will contribute to the deliverable D5.4, due in Month 44 (M44). Different components will contribute various functionalities across each release, as outlined below. Once a component delivers functionality in a given release, it is expected that this functionality will remain available in subsequent releases, with only minor updates, rather than major changes, made over time. This time plan helps streamline integration for future steps and prepares the releases to be used by the project's pilots.

Table 31: Integration execution time plan (participation of components in releases)

Task(s)	Component Name	Partner	Participation in Alpha Release (M24)	Participation in Beta Release (M30)	Participation in Final Release (M44)
T4.3, T4.4	**BNB**	CSCY	Servers in clinical sites, data processing pipelines (harmonization, curation), insertion of	X	Pseudonymization

			retrospective data, descriptive data generation		
T4.3, T4.4	**Citizen BNB**	CSCY	Server in cloud	X	X
T4.1	**IMS**	INTRA	Real data retrieval	Synthetic data retrieval, enriched (mixed real and synthetic) data retrieval, statistics aggregator	X
T5.4	**RISKA**	AINIGMA		X (full functionality)	X
T5.4	**RECENG**	AINIGMA	X (full functionality)	X	X
T5.5	**SDG**	NVCR		Synthetic data generation, enriched (mixed real and synthetic) data generation	X
T5.2	**DASH**	TMA	User login, real data querying, recommendation listing, Node Bundles monitoring, link to active health app	Risk assessment, synthetic and enriched data querying	X
T4.3	**NGW**	TMA	X	X	X
T5.2	**LOGS**	TMA	X	X	X
T5.2	**METRICS**	TMA	X	X	X
T5.2	**LOC**	TMA	X	X	X
T5.2	**ANA**	TMA	X	X	X

T3.2	**COST**	i2G			X
T5.3	**SMS**	STS	X (full functionality)	X	X
T5.6	**ACT**	TMA	X (full functionality)	X	X
T5.6	**SERG1**	HUA	X (full functionality)	X	X
T5.6	**SERG2**	HUA	X (full functionality)	X	X
T5.6	**SERG3**	HUA	X (full functionality)	X	X
T5.3	**AUTH**	STS	X	X	X
T5.6	Marketplace (dependency for ACT, SERG)	HUA	X	X	X
T7.1	Knowledge Hub (dependency for ACT, SERG)	MARTEL	X	X	X
T7.3	Community Network (dependency for ACT, SERG)	NUCLIO	X	X	X

Conclusions

This deliverable provides a detailed technical architecture of the BIO-STREAMS platform, building upon the conceptual foundations established in Deliverable D2.3. Several updates have been introduced, including modifications to the original conceptual architecture and the addition of new components, particularly those designed to support the school pilots, observability, and the cost model. The logical views of these components have been thoroughly presented, reflecting the platform's growing complexity and capability.

Adopting the 4+1 architectural model and building on D2.3, we have outlined the process view of the platform, detailing all major processes and their corresponding sequence diagrams to capture the dynamic interactions between components. The deployment view is also presented, encompassing hardware specifications, security measures, and the architecture's overall infrastructure. Furthermore, this deliverable elaborates on the CI/CD tools and the software development workflow, ensuring a clear path for continuous integration and deployment. Finally, we have introduced the platform's integration methodology, supported by an integration matrix defining the communication protocols between components, and an integration execution time plan outlining the contributions of each component in the Alpha, Beta, and Final releases. These comprehensive updates are designed to ensure a robust and scalable platform, aligning with the evolving needs of the pilot projects.

References

[1] "Seq - The self-hosted search, analysis, and alerting server," [Online]. Available: https://datalust.co/seq.

[2] "OpenTelemetry: High-quality, ubiquitous, and portable telemetry to enable effective observability," [Online]. Available: https://opentelemetry.io/.

[3] "Prometheus: From metrics to insight. Power your metrics and alerting with the leading open-source monitoring solution.," [Online]. Available: https://prometheus.io/.

[4] "Grafana: Query, visualize, alert on, and understand your data no matter where it's stored. With Grafana you can create, explore, and share all of your data through beautiful, flexible dashboards.," [Online]. Available: https://grafana.com/.

[5] "matomo: Google Analytics alternative that protects your data and your customers' privacy," [Online]. Available: https://matomo.org/.

[6] "FIWARE Reference Architecture and Components.," FIWARE Foundation, 2023. [Online]. Available: https://www.fiware.org/catalogue/.

[7] *Refinement of the eHealth European Interoperability Framework (ReEIF),* 7th Meeting of the eHealth Network, April 28, 2015: eHealth Network.

[8] *Refined eHealth European Interoperability Framework. Adopted by consensus by the eHealth Network,* Brussels, 23 November 2015: eHealth Network.

[9] "OpenAPI Specification," OpenAPI Initiative, [Online]. Available: https://spec.openapis.org/oas/latest.html.

[10] "Swagger.io - "What Is OpenAPI?," [Online]. Available: https://swagger.io/docs/specification/about/.

[11] "OpenAPI Specification Version 3.0.3," OpenAPI Initiative, [Online]. Available: https://spec.openapis.org/oas/v3.0.3.

[12] "GitHub: Let's build from here - The world's leading AI-powered developer platform.," [Online]. Available: https://github.com/.

[13] "Jenkins - Build great things at any scale - he leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.," [Online]. Available: https://www.jenkins.io/.

[14] "Docker: Accelerated Container Application Development | Docker is a platform designed to help developers build, share, and run container applications. We handle the tedious setup, so you can focus on the code.," [Online]. Available: https://www.docker.com/.

[15] "Docker Compose is a tool for defining and running multi-container applications. It is the key to unlocking a streamlined and efficient development and deployment experience.," [Online]. Available: https://docs.docker.com/compose/.

[16] "Harbor is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted.," [Online]. Available: https://goharbor.io/.

[17] "Portainer is your container management software to deploy, troubleshoot, and secure applications across cloud, datacenter, and Industrial IoT use cases.," [Online]. Available: https://www.portainer.io/.

[18] "Keycloak: Open Source Identity and Access Management | Add authentication to applications and secure services with minimum effort. Keycloak provides user federation, strong authentication, user management, fine-grained authorization, and more.," [Online]. Available: https://www.keycloak.org/.

[19] "WEBLATE: Web-based continuous localization - Copylefted libre software, used by over 2,500 libre software projects and companies in over 165 countries.," [Online]. Available: https://weblate.org/el/.

[20] "GNU gettext is an important step for the GNU Translation Project, as it is an asset on which we may build many other steps. This package offers to programmers, translators, and even users, a well integrated set of tools and documentation.," [Online]. Available: https://www.gnu.org/software/gettext/.