

# retention

SMART HEART FAILURE MANAGEMENT

## Horizon 2020 Project RETENTION

**“HEART FAILURE PATIENT MANAGEMENT AND INTERVENTIONS USING CONTINUOUS PATIENT MONITORING OUTSIDE HOSPITALS AND REAL WORLD DATA”**

**Research and Innovation Action  
H2020-SC1-BHC-2018-2020  
GA 965343**

**Duration: 48 months from 01/05/2021**

**Coordinator: Institute of Communication and Computer Systems**

Deliverable ID.:	<b>D7.2</b>
Deliverable title:	<b>Report on integrated RETENTION platform v1</b>
Planned delivery date:	31/01/2023
Actual delivery date:	29/09/2023
Deliverable leader:	SIEMENS SRL
Contributing partners:	DM, ICCS, FORTH, STS, AEGIS, SIESRL, i2G
Dissemination Level:	<input checked="" type="checkbox"/> PU = Public
	<input type="checkbox"/> CO = Confidential
	<input type="checkbox"/> CI = Classified



This project has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 965343.

This deliverable reflects only the authors’ view and the Commission is not responsible for any use that may be made of the information it contains.



## Document information and history

### Deliverable description (from DoA)

A report documenting the integrated platform's (a) functionalities and (b) any amendments made to its architecture and design, (c) the outcomes of testing the platform and (d) installation, deployment and usage guidelines for the platform.

Version N.	Date	Author [Person and Organisation]	Reviewer [Person and Organisation]	Milestone*	Notes
V. 0.1	10/01/2023	I. E. Nicolae, G. Danciu (SIESRL)		ToC	General document structure
V. 0.2	26/01/2023	I. E. Nicolae, G. Danciu (SIESRL)		1st draft version	Initial Contributions
V. 0.3	04/02/2023	F. Picioroaga, I. E. Nicolae, G. Danciu (SIESRL)		2nd draft version	First round of contributions
V. 0.4	10/02/2023	F. Picioroaga, I. E. Nicolae, G. Danciu (SIESRL), M. Kokkonidis (AEGIS), D. Zakaki, G. Del Toro, (DM)		3rd draft version	2nd round of contributions
V. 0.5	10/04/2023	I.E. Nicolae, G. Danciu (SIESRL), M. Roumpi, T. Pardalis (FORTH)			Updated contributions, text editing
V. 0.6	13/07/2023	I.E. Nicolae, G. Danciu (SIESRL), M. Roumpi, T. Pardalis (FORTH)			Updated contributions, text editing
V. 0.7	29/07/2023	I.E. Nicolae (SIESRL), M. Roumpi (FORTH)	M. Kokkonidis (AEGIS)	1 <sup>st</sup> internal review	Restructuring after feedback, new contributions
V.0.8	1/08/2023	M. Kokkonidis (AEGIS), N. Vasileiou (ICCS), I. Vezakis, L. Koumakis (STS), G. Danciu, I.E. Nicolae (SIESRL),			New contributions
V. 0.91	1/09/2023	G. Danciu, I.E. Nicolae (SIESRL), G. Del Torro, A. Deloglou (DM), L. Koumakis (STS), M. Kokkonidis (AEGIS)	I. Kouris (ICCS)	2 <sup>nd</sup> internal review	New contributions, corrections after review
V. 0.92	13/09/2023	G. Danciu, I.E. Nicolae (SIESRL), T. Pardalis (FORTH)	M. Colombo (i2G)	3 <sup>rd</sup> internal review	Corrections after review
V. 0.93	21/09/2023	I.E. Nicolae (SIESRL), M. Kokkonidis (AEGIS)			Comments addressed after review
V. 0.94	22/09/2023		N. Vasileiou (ICCS)	Review and corrections	



Version N.	Date	Author [Person and Organisation]	Reviewer [Person and Organisation]	Milestone*	Notes
V. 0.95	25/09/2023		L. Koumakis (STS)	Deliverable approved by the TC	
V. 0.96	26/09/2023		M. Haritou (ICCS)	Final Review	Format corrections applied
V. 1.0	29/09/2023		M. Haritou (ICCS)	Deliverable Submission	Deliverable approved and submitted by the PC



## Table of Contents

Deliverable description (from DoA).....	2
List of Abbreviations.....	7
Table of Figures.....	9
List of Tables.....	10
1. Executive Summary.....	11
2. About this Document.....	12
2.1. Role of deliverable.....	12
2.2. Relationship to other RETENTION deliverables.....	12
2.3. Structure of the document.....	13
3. RETENTION Platform Overview.....	14
3.1. Architecture.....	14
3.1.1 CSB Communication Pathway.....	15
3.1.2 GIC Communication Pathway.....	15
3.2 Patient Edge (PE).....	17
3.2.1 Mobile App.....	17
3.2.2 Local Home Gateway.....	17
3.3 Clinical Site Backend (CSB) layer.....	18
3.3.1 CSB Dashboard.....	18
3.3.2 Big Data Analysis Executor.....	19
3.3.3 Decision Support System.....	20
3.3.4 FHIR and non-FHIR real-world Data Repositories.....	21
3.3.5 Security Component - Sphynx.....	21
3.4 Global Insights Cloud (GIC) layer.....	23
3.4.1 GIC Dashboard.....	23
3.4.2 Big Data Analytics Engine.....	23
3.4.3 Model Specification Tool.....	23
3.4.4 Disease Insights.....	25
3.4.5 Decision and Policy Support.....	25
3.4.6 FHIR and non-FHIR Data Repositories.....	25
3.4.7 Data and model sharing.....	25
3.4.8 Repository for Models.....	26



---

3.4.9 Security Component .....	27
4. Installation, deployment, and integration.....	28
4.1. Overall platform installation, set-up and deployment.....	28
4.1.1 GIC/CSB Dashboard .....	28
4.1.2 Model Specification Tool, BDA Engine and Disease Insights.....	29
4.1.3 GIC/CSB FHIR and non-FHIR Data Repositories.....	29
4.1.4 Decision Support System.....	29
4.1.5 Local Home Gateway.....	29
4.1.6 Data and Model Sharing, Decision and policy support .....	30
4.1.7 Mobile App .....	30
4.2. Integration.....	30
4.2.1 Platform integration .....	30
4.2.2 Integration technology .....	32
4.2.3 Overall system integration .....	32
5. Technical Testing & Quality Assurance.....	36
5.1 Technical Testing .....	36
5.1.1. Testing at the level of each module .....	37
5.1.2. Integration testing.....	37
5.1.3. Use case testing, end-to-end testing.....	38
5.1.4. UI testing .....	42
5.1.5. Security testing.....	42
5.1.6 Performance testing.....	43
5.2 Quality Assurance.....	43
5.2.1 Robust development process.....	45
5.2.2 Collect and incorporate user feedback .....	45
6. Next work and strategy .....	46
6.1 Further development .....	46
6.2. Testing planning .....	46
6.3. Quality Assurance planning.....	47
6.3.1 Elaborate the testing procedures.....	47
6.3.2 Continuously collect and incorporate user feedback.....	47
6.2.3 Continuous improvement of the platform.....	47
7. Conclusions.....	48



---

Appendix A. Usage guidelines .....	49
A.1. CSB Dashboard and GIC Dashboard .....	49
A.2. FHIR Implementation guide .....	49
A.3. Data and Model Sharing .....	49
A.4. Model Specification Tool, Big Data Analytics Engine, Disease Insights.....	50
A.5. Mobile App .....	59
A.6. Local Home Gateway.....	59
Appendix B. Setup instructions .....	60
B.1. MST, BDA, DI: Setup installation steps .....	60
B.1.1 GIC layer .....	60
B.1.2 CSB layer .....	61
B.1.3 Accessing the REST API.....	62
Appendix C. Docker-compose files .....	64
C.1 MST, BDAE, DI docker-compose files .....	64
C.1.1. MST, BDAE, DI Docker-compose on GIC side .....	64
C.1.2. BDAEE Docker-compose on CSB side .....	68



## List of Abbreviations

Essential abbreviations used in the document:

HF	Heart Failure
AUTH	Authorization
BDAE	Big Data Analytics Engine or BDA Engine
BDAEE	BDA Engine Executor
Carer	Caregiver
CCM	Clinical Case Manager
CI/CD	Continuous Integration/Continuous Deployment
CSB	Clinical Side Backend (part of the system)
CSBDB	CSB Dashboard
cSPAP	continuous Security and Privacy Assurance Platform
CSS	Cascading Style Sheets
CSV	Comma-separated values
DI	Disease Insights
DoA	Description of Action
DPS	Decision & Policy Support
DSS	Decision Support System
e-CRF	electronic Case Report Form
ECG	Electrocardiogram
EDGE	Microsoft Edge
GDPR	General Data Protection Regulation
GET	HTTP GET method
GIC	Global Insights Cloud (part of the system)
GUI	Graphical User Interface
HT	Heart Transplant
Carer	Caregiver
IMEI	International Mobile Equipment Identity
IoT	Internet of Things
JS	JavaScript
JSON	JavaScript Object Notation
KPI	Key Performance Indicator



LHG	Local Home Gateway
LVAD	Left Ventricular Assisted Devices
ML	Machine Learning
MST	Model Specification Tool
OS	Operating System
PD	Patient Devices
PD	Patient Devices
PII	Personal Identifiable Information
POST	HTTP POST method
PQ/PR, QRS, QT	ECG markers
QA	Quality Assurance
vPoP	virtual Pilot of Pilots
RBAC	Role-based Access Control
REST	Representational State Transfer
RPI / RASP-PI	Raspberry Pi
SUS	System Usability Scale





## Table of Figures

Figure 1. Architecture (revised diagram, updated after D3.2 “RETENTION Architecture”) .....	15
Figure 2. The CSB@SecurityComponent replaces Pseudo-Id1 to Pseudo-Id2 by which pseudonymised data record is stored in CSB@Repository and vice versa. In a similar way, CSB@SecurityComponent replaces Pseudo-Id1 to Pseudo-Id3 by which pseudonymised data record is stored in GIC@Repository.....	22
Figure 3. The containers that comprise the Security Components, their connections, and their dependencies. ....	35
Figure 4: Sample of unit test from the security component .....	43
Figure 5. Roles and their interaction within ML components.....	51
Figure 6. Log in for the Model Specification Tool.....	51
Figure 7. Create a new Notebook.....	52
Figure 8. Sample ML Python code .....	52
Figure 9. Running the notebook .....	53
Figure 10. Log in screen for GIC Disease Insights .....	53
Figure 11. List of the performed notebook training experiments.....	54
Figure 12. Visualise the model’s performance .....	55
Figure 13. Code sample for model prediction.....	56
Figure 14. Sample clinician screen for ML activity .....	58
Figure 15. The project structure of the model specification tool .....	60
Figure 16. The successful start of the docker image .....	61
Figure 17. Details of the started containers in the analytical stack on GIC.....	61
Figure 18. Project structure for the CSB side .....	61
Figure 19. Validation of images running on CSB .....	62
Figure 20. Details of the started containers in the analytical stack on CSB .....	62



## List of Tables

Table 1: Notifications/Interventions examples .....	20
Table 2: End-user scenarios prerequisites: set-up and data. ....	39
Table 3: Example of VPoP demo testing actions for the functional assessment. ....	44



## 1. Executive Summary

This document reports on the first version of the RETENTION platform, an integrated healthcare platform designed to monitor and provide potential interventions for chronic Heart Failure (HF) patients. The platform uses a range of data including medical, clinical, physiological and behavioural data to detect patterns of disease progression and offer personalized interventions. This is possible because the platform includes big data analytics and machine learning techniques.

The report provides a detailed overview of the platform's functionalities, amendments made to its architecture and design, its installation, deployment, and usage guidelines, and outcomes of testing and validating the platform. The platform is closely related to various other deliverables that define the requirements, architecture, data model, analytics, and decision-making mechanisms of the RETENTION solution.

The RETENTION platform is divided into three layers: the Global Insights Cloud (GIC), the Clinical Site Backend (CSB), and the Patient Edge (PE). The PE layer involves the Mobile App and the Local Home Gateway (LHG) that collect and pre-process data from home-based sensors and medical devices. The CSB layer contains components responsible for monitoring patients' health and wellbeing at clinical centres. The GIC layer is responsible for data analysis and decision-making.

Two amendments were made to the architecture: the creation of a full test environment for testing and staging and the introduction of reverse proxy software to conserve IP addresses and control the back end.

The document lays out future development and integration strategies, emphasizing on continuous testing, adherence to data privacy principles, and usability improvements for clinicians.



## 2. About this Document

This document provides a comprehensive description of the first version of the integrated RETENTION platform, a state-of-the-art healthcare tool designed to monitor and deliver targeted interventions to patients with chronic Heart Failure (HF). It captures the platform's functionalities, changes in its design and architecture, detailed installation, deployment, and usage guidelines, and outlines the results of initial testing and validation processes.

### 2.1. Role of deliverable

This report escorts the first version of the integrated RETENTION platform, involving all the first release of components of WP4, WP5 and WP6 developed until M21, as demonstrated in D7.1 "Integrated RETENTION platform v1". Its purpose is to document the functionalities of the 1st version of the integrated platforms and any amendments made to its initially proposed architecture and design, further detail the installation, deployment, and usage guidelines for the platform, along with the outcomes of testing and validating the platform.

### 2.2. Relationship to other RETENTION deliverables

Before describing the structure of the document, first a more general context within the RETENTION deliverables will be provided.

This deliverable is closely related to:

- D3.1 "RETENTION Requirements" which is the first deliverable of "WP3 - RETENTION Clinical and System Requirements & Platform Design". This deliverable describes the work that was done to define the requirements of the RETENTION solution, covering the patient clinical management, user, and platform requirements.
- D3.2 "RETENTION Architecture" which presents the detailed architecture of the RETENTION platform in order to coordinate the implementation of the various technical components that constitute it.
- D4.1 "The RETENTION Data Model" and D4.2 "Data Management enabling mechanisms" that will use the RETENTION data model framework to design and deploy data repositories, store the medical and non-medical data, and implement data transfer and handling mechanisms.
- D5.1/D5.2 "RETENTION Analytics & Decision-making enabling mechanisms v1/v2" that will consider the RETENTION data model by providing an open data sharing specification that will form the basis for clinical and RWD data sharing from and to the RETENTION platform.
- D6.1 "RETENTION Interfacing, Device Federation and Visualisation components v1" which includes details on the Mobile Application, Gateway, and smart devices federation, as well as CSB and GIC dashboard and visualisation components.
- D6.2 "RETENTION Security & Privacy by design enabling mechanisms v1" which covers the fundamentals, the development, the modelling, the configuration, and execution of mechanisms to guarantee the security and privacy of data held in the RETENTION platform.
- D7.1 "Integrated RETENTION platform v1 demonstrator", involving the first release of components covering the first version of the RETENTION platform.



### 2.3. Structure of the document

The current deliverable provides the integration status of the RETENTION components. The purpose of this version is to provide an overview of the developed modules, the installation and deployment of these components, the technical testing methods, and the future steps. The structure of the D7.2 “Report on integrated RETENTION platform v1” is as follows:

Section 1 presents the purpose of this deliverable.

Section 2 presents the overview of this deliverable, the relationship to other RETENTION deliverables.

Section 3 presents an overview of the developed components, the interaction between them and implementation status.

Section 4 describes the installation steps, integration effort, and deployment steps to create the current version of the RETENTION platform.

Section 5 presents the technical testing of each component of the platform.

Section 6 describes the strategy and future developments of the integrations and testing.



## 3. RETENTION Platform Overview

The RETENTION Platform represents an innovative, multi-layered architecture designed to help healthcare domain through data-driven insights and intelligent decision-making.

The primary goal of the RETENTION project is to develop an integrated platform that provides clinical monitoring and interventions to chronic Heart Failure (HF) patients. The platform gathers and analyses health-related data including medical, clinical, physiological, behavioural, psychosocial, and real-world data, with further analysing the Heart Failure study participants' activities and progression. The data will be continuously collected, processed, and interpreted, cross checked and validated against clinical literature.

At its core, the platform is structured around the Clinical Side Backend (CSB) and the Global Insights Cloud (GIC). The CSB is focused on managing all aspects related to individual clinical sites, including the gathering of patient data from various smart devices and sensors installed in their homes. This information is stored in a pseudonymized format to ensure privacy and security.

The GIC layer acts as a centralized repository and analytics engine that accumulates and processes data from various CSBs. The Big Data Analysis Engine (BDAE) within the GIC takes care of large-scale data handling and feeds machine learning models for advanced analysis.

In this chapter we will present the general architecture of the platform and describe without going into details, the main components that make the platform work. These are the CSB and GIC, along with their subcomponents.

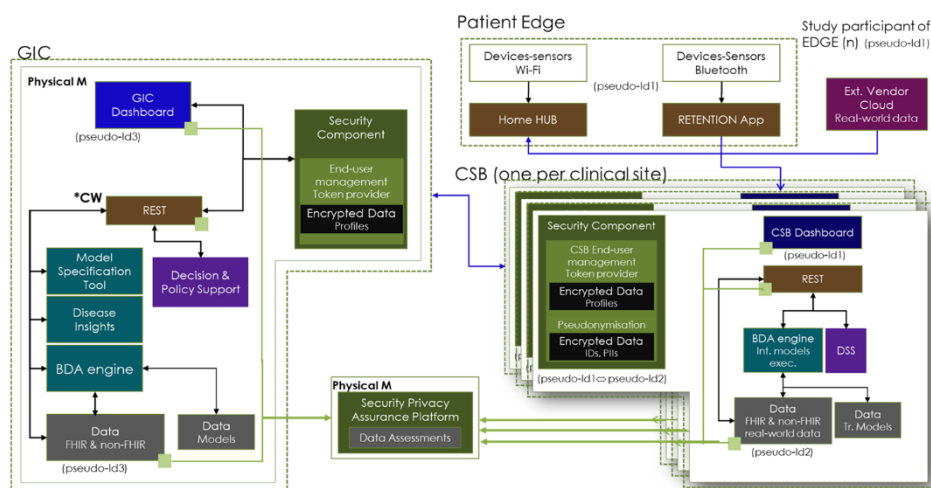
### 3.1. Architecture

Conceptual and detailed architecture of the RETENTION project, that adheres to the identified end-user requirements (as presented in D3.1 "RETENTION requirements" and D3.2 "RETENTION Architecture"). Following an iterative process, the main outcome of T3.3 "RETENTION Platform Architecture & Design" has been the overall architecture of the RETENTION platform including detailed component descriptions of the main blocks of the architecture.

As described in D3.2 "RETENTION Architecture", the architecture consists of 3 layers:

- Global Insights Cloud (GIC),
- Clinical Site Backend (CSB) and
- Patient Edge (PE).

The diagram from Figure 1 presents an overview of the RETENTION Architecture and actors operating within each layer.



**Figure 1. Architecture (revised diagram, updated after D3.2 “RETENTION Architecture”)**

The RETENTION solution comprises of three main components: the GIC layer, the CSB layer, and the Patient Edge (PE) layer. The GIC layer includes sub-components such as the GIC Dashboard, Federated RW Data Repository, Repository for Models, BDA Engine, Model Specification Tool, Disease Insights, Decision & Policy Support, Security Component, and GIC Rest API. The CSB layer includes sub-components such as the CSB Dashboard, FHIR & Non-FHIR Repository, BDA Engine Models Executor, Decision Support System, Chat System, Security Component, and CSB Rest API. The PE layer includes the Mobile App and Local Home Gateway. Additionally, there is a security component that ensures privacy and data protection for all operations.

The RETENTION architecture has undergone minor revisions since the submission of D3.2 “RETENTION Architecture”. These changes specifically affect the deployment strategy of the Security and Privacy Assurance (SPA) platform. Instead of deploying it multiple times at each CSB, the new approach involves a centralized deployment, executed only once.

This decision was motivated by the aim to enhance resource management. The SPA platform consumes substantial hardware resources, and a single instance has demonstrated the capacity to efficiently handle the workload of all CSBs and the GIC, ensuring optimal resource utilization.

### 3.1.1 CSB Communication Pathway

**Mobile App and Home Gateway:** Both components make RESTful requests to the CSB Security Component to receive access tokens.

**CSB Dashboard:** The front-end dashboard also communicates with the CSB Security Component for access tokens and subsequently interacts with the FHIR/Non-FHIR Repositories of the CSB Backend.

**CSB Backend:** Upon successful authorization through the Security Component, the backend can be accessed to fetch or send data to the FHIR/Non-FHIR Repositories.

### 3.1.2 GIC Communication Pathway

**Mobile App and Home Gateway:** Both components make RESTful requests to the GIC Security Component to receive access tokens.



Local Home Gateway: Sends RESTful requests to the GIC Security Component for access tokens, allowing it to send data to the CSB backend services.

The GIC and CSB Dashboards allow for data input and usage of services and communicate with other components through REST interfaces. REST services also provide communication between the BDA engine, Data Repository, and different repositories located in GIC or CSB. Each Clinical Site Backend interacts with the Local Home Gateway and Mobile Application through REST interfaces. The Local Home Gateway collects data from home-based sensors, while the Mobile Application handles medical and non-medical device information. This information is pre-processed and sent to the Clinical Site Backend in compliance with GDPR regulations through REST services. Interactions between these components, as well as authentication at the Dashboards, are provided by the Security Component, which also handles end-user account information.

A low impact amendment was introduced to the architecture and the implementation of the system (provided in D3.2 “RETENTION Architecture”), considering a reverse proxy software meant to conserve on IP addresses and have better control on the back end.

The RETENTION architecture adheres to the “Privacy by Design” principle and allows study participants' data to be used for further utilisation in the context of big data analytics, while ensuring data privacy and protection.

In the following, an overview of the components is provided, organised in terms of where they operate (Patient Edge, CSB Layer, GIC Layer). For each component a short description of their functionality is provided.





## 3.2 Patient Edge (PE)

The Patient Edge focuses on two pivotal components of the RETENTION Platform that empower both patients and healthcare providers with this real-time, data-driven approach: the Mobile App and the Local Home Gateway.

The Patient Edge components, including the RETENTION Mobile App and Home Gateway, communicate with the CSB and GIC backends via RESTful requests. Access tokens, obtained from respective security components, authorize these interactions, ensuring that only approved entities can access the sensitive data.

### 3.2.1 Mobile App

The Mobile App setup consists of the RETENTION application (to be installed on the Android device provided by the project), a Garmin smartwatch and several measurement devices from OMRON (oximeter, pressure gauge, weight scale). The application's main goal is measuring patient health data in the following ways:

- Constant measurement of heart rate, calorie burn, steps taken/distance travelled and sleep quality through the Garmin-provided smartwatch. The data is synchronized with the application and transmitted to the CSB at least once a day.
- Measurement of oxygen saturation, blood pressure and weight by interfacing with OMRON devices via Bluetooth. After pairing the devices with their android phone, patients take daily measurement of these metrics, and the data is transmitted back to the CSB.
- Manual entry of temperature measurements. Patients manually take their temperature, then enter it into the application via a specific manual entry screen.
- Measurements of the VAD device Speed (RPM), Flow (LPM), Pulse Index, Power (W) and parameters regarding Duration, Alarm and Actions for LVAD patients, sent along with photos of the devices' screens captured by patients/carers for validation purposes.

The application then temporarily stores the data locally and transmits it to the CSB through the security component and the system API, as soon as an internet connection is made available.

### 3.2.2 Local Home Gateway

The Local Home Gateway consists of a Raspberry Pi microcomputer, a temperature and humidity sensor, and an external weather service. Its main functions are to measure temperature, humidity, external temperature, external humidity, and air pollution levels, store the data locally, transmit it to the Clinical Site Backend (CSB) through the security component via a non-FHIR API, handle data loss, allow remote access for technical support, identify and report low sensor battery levels, and transfer data securely using specific credentials. It operates automatically and can acquire external weather and pollution data using zip codes, municipalities, and areas.

The Local Home Gateway (as described in D6.1 "RETENTION Interfacing, Device Federation and Visualisation components v1" in detail), aims at gathering non-FHIR data in real time each hour, storing the raw data locally, converting them to the necessary non-FHIR data structures and then efficiently and securely uploading them to the Clinical Site Backend (CSB) DB. The Local home Gateway comprises a Raspberry Pi mini computer that acts as an EDGE computer, a temperature and humidity sensor (Xiaomi Mijia) responsible for measuring the indoor conditions and a free online weather service (open weather service map) to measure accurately the outdoor weather conditions (external temperature, external humidity and pollution levels of area of interest).



The main functions of the Local Home Gateway are:

- **Real-Time Measurement:** Captures temperature, humidity, external temperature, external humidity, and air pollution index values
- **Hourly Data Gathering:** Collects all the required data values each hour, running 24 hours a day during the pilot.
- **Local Storage:** Stores the captured data locally for later analysis or retrieval.
- **Data Transmission:** Transmits the locally stored data to the Clinical Site Backend (CSB) via a non-FHIR API.
- **Data Loss Handling:** Has built-in mechanisms for detecting and handling data loss and employs workaround solutions to minimize data loss.
- **Remote Access:** Allows for remote access capabilities to support technicians during the pilot process.
- **Low Battery Reporting:** Identifies and reports if the sensor battery levels are low.
- **Security:** Ensures secure data transmission through specific credentials, ensuring no compromises in data integrity.
- **Weather and Pollution Data:** Acquires external weather and pollution data based on zip codes, municipalities, and areas.

The functionalities supported by the EDGE Computer (Raspberry Pi) are:

- **Real-Time Measurement:** The Raspberry Pi is responsible for interfacing with the Xiaomi sensor to capture temperature and humidity.
- **Local Storage:** Utilizes the Raspberry Pi's local storage capabilities to hold onto the captured data.
- **Data Transmission:** Uses the Raspberry Pi to transfer data securely to the Clinical Site Backend (CSB) via non-FHIR API.
- **Remote Access:** The Raspberry Pi allows for remote access by technicians for troubleshooting and support.
- **Security:** The Raspberry Pi works in conjunction with a security component to ensure that data is transmitted securely using specific credentials related to the local server, `userId`, `packageId`, and `organizationId`.
- **Automatic Updates:** The Raspberry Pi, being an open-source platform, facilitates automatic software updates.

### 3.3 Clinical Site Backend (CSB) layer

The RETENTION CSB layer comprises a collection of components responsible for monitoring the health and wellbeing of patients enrolled in the program at one clinical centre. Data collected from patients' home (through the patient's mobile phone and home gateway) are made available to the clinical centre's CSB, in real-time. In the next platform version, verified AI models based on these data alongside the clinical data (EHR) will enable predictive local analytics offering additional information to clinicians and, as a result, also potentially enhancing their decision making.

#### 3.3.1 CSB Dashboard

The Clinical Site Backend (CSB) Dashboard provides a GUI for clinicians to find, process, and analyse the data gathered by the RETENTION platform for all patients enrolled in the RETENTION study, providing a e-CRF (electronic Case Report Form) functionality to clinicians in the case of patients assigned to the control group,



reserving advanced functionality, what RETENTION technology brings to clinical practice, for intervention group patients.

The CSB Dashboard (CSBDB) allows clinicians to create patient records and track their basic information and standard clinical monitoring data in the course of the RETENTION study. However, for patients in the intervention group they also have access to additional functionalities:

- RETENTION patient monitoring device data visualisations allowing clinicians to remotely monitor a number of aspects of their patients' health as well as their adherence to their medication regime
- RETENTION alarms drawing clinicians' attention to any developments that may require their intervention
- RETENTION AI-based functionality offering decision support to clinicians on a number of issues they identify as important in-patient care.

The CSB Dashboard also plays a role in supporting RETENTION-based monitoring via smart connected devices and sensors, via allowing technicians to register the smart data transmitting devices provided to each patient (smartphone, Raspberry Pi-based Home Gateway) via their unique identifiers (IMEI, LAN MAC Address respectively), update the relevant record if the need arises (e.g. a device is lost or damaged), and monitor that patient devices (PD) are functioning properly and sending data to the corresponding CSB backend.

The CSB functionality is further described in its User Guide provided as Appendix 1.

### 3.3.2 Big Data Analysis Executor

The entire AI-based functionality that works as an assistant for clinicians, will be implemented into the RETENTION platform version 2.

The patient data transmitted to GIC FHIR repository feeds the Big Data Analytics Engine, at the GIC side, that will be used to create meaningful AI models. The trained models created within the GIC side, are transmitted to all CSB instances. The Big Data Analytics Engine Executor supports the applicability of the previous trained ML models to predict the possible future outcomes of a patient's (health or wellbeing) indicator. As such, this component provides means of receiving the latest variant of a ML model and applying it to predict patients' possible evolution of an indicator. An example of such an indicator as defined by the endpoints in D8.2 "RETENTION Clinical Trial Protocol and Evaluation Framework", is related to cardiovascular mortality.

After these models are trained within the Jupyter Notebook environment using the Model Specification Tool and Big Data Analytics Engine from the GIC layer, they need to be exposed so they can be used by the Clinical Side Backend (CSB). To achieve this, the Jupyter Gateway serves as an intermediary, exposing the trained models via a RESTful API. This allows the CSB to consume these models for making predictions. The CSB server retrieves the trained model from the Global Insights Cloud (GIC) server using the MLFlow's REST API and the Jupyter Gateway API. The model can then be used to make predictions on the patient's data collected from various sources, including medical analyses and wearable devices.

The Jupyter Gateway ensures that these models are accessible and ready for use, enabling clinicians to interact with them through the CSB dashboard. Here, they can choose the models they want to use to get predictions for their patients based on different scenarios.

After the call to the trained model is made through the Jupyter Gateway, the prediction results are then displayed on the dashboard. This allows clinicians to analyse the information and make data-informed decisions for their patients, supplementing their existing clinical knowledge.

In summary, the Jupyter Gateway plays a vital role in the Big Data Analysis Executor's functionality by enabling the exposure and consumption of trained machine learning models for prediction.

### 3.3.3 Decision Support System

The Decision Support System (DSS) is responsible for generating patient notifications and interventions based on rules defined by the clinicians (D8.2 "RETENTION Clinical Trial Protocol and Evaluation Framework"), see example below. The DSS continuously monitors medical data stored in the FHIR repository and compares single values or calculated averages against the provided thresholds. Notifications are created for both study groups; however, the clinicians will be informed only for notifications from the intervention group via the dashboard.

**Table 1: Notifications/Interventions examples**

Clinical characteristics	Type of Notification	Definition of Notification	Type of Intervention	Clinical Intervention
<b>Weight:</b>	<i>Low level:</i>	Gain >2kg in one day or gain > 3 Kg in even days	1. Dismiss notification - measurement error - not clinically relevant 2. Telephone visit 3. Schedule in-person visit	- Reinforce dietetic plan - Increase diuretic dose
	<i>Critical level:</i>	Gain ≥3kg in 2 days	1. Dismiss notification - measurement error - not clinically relevant 2. Telephone visit 3. Schedule in-person visit	- Reinforce dietetic plan - Increase diuretic dose - Check if IV diuretic is needed - Consider visiting the emergency department (ED)
<b>Blood pressure:</b>	<i>Low level:</i>	Systolic Blood Pressure (SBP) < 100 mmHg and a 10 mmHg reduction from the average of the previous 3 measurements) or SBP>140mmHg	1. Dismiss notification - measurement error - not clinically relevant 2. Telephone visit 3. Schedule in-person visit	- Check for triggers: Temperature, hypovolemia... - Check if Hypovolemia (sweat, diarrhoea) - Consider reducing dose of diuretics, ARBs, B-blockers, MRAs, ARNIs etc
	<i>Critical level:</i>	Systolic Blood Pressure (SBP)< 90 mmHg and a 10 mmHg reduction from the average of the previous 3 measurements mmHg or SBP>150mmHg	1. Dismiss notification - measurement error - not clinically relevant 2. Telephone visit 3. Schedule in-person visit	- Check for triggers: Temperature, hypovolemia... - Check Temperature - Check if Hypovolemia (sweat, diarrhoea) - Consider adjust dose of ARBs, B-blockers, MRAs, ARNIs, etc. - Consider visiting the emergency department (ED)



### 3.3.4 FHIR and non-FHIR real-world Data Repositories

As described above, the FHIR repository is responsible for the storage of all medical data (patients, caregivers, conditions, observations, questionnaires, medications, visits and hospitalisations). Sensitive data regarding patients and caregivers are only stored in the security component, thus making data residing in the FHIR repository anonymised. Most of the data is manually imported by the clinicians through the dashboard, while some real time measurements by smart devices are periodically transmitted via the mobile app. The structure of the FHIR JSON files alongside with the mapping to international ontologies and terminology systems (SNOMED, LOINC, ICD) are included in the FHIR implementation guide (in Appendix A.2).

The non-FHIR repository is used for the storage of environmental measurements transmitted via the Local Home Getaway. It also stores the patient notifications-interventions generated by the Decision Support System.

Interoperability among the two repositories and every other component is mediated by the Security Component.

A different instance of a FHIR and non-FHIR repository will be used for each clinical centre.

### 3.3.5 Security Component

The RETENTION project is motivated by the need for personal data management and services that comply with the General Data Protection Regulation (GDPR), which has already imprinted during the requirements stage. To support the secure management and transferring of such information without compromising the privacy of the data-subjects, the whole solution has in place secure ways to handle, digest, distribute, and present (i.e., via the Dashboard or the Mobile application) information, to the correct set of 'eyes' (i.e., end-users with appropriate access rights based on Role-based Access Control – RBAC), without compromising usability or speed. The Security Component, which is present in both GIC and CSB instances, facilitates the following:

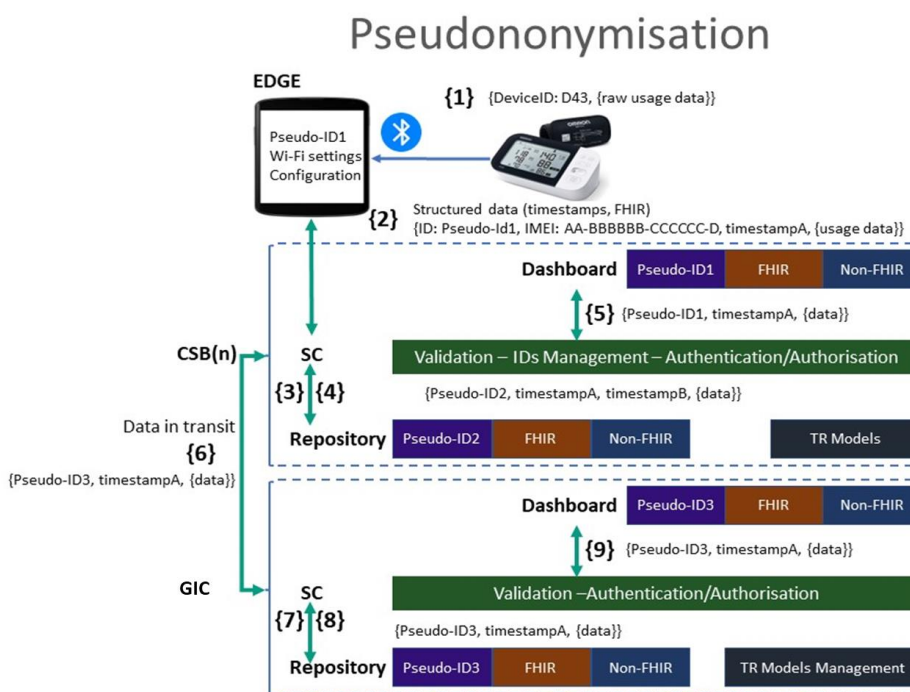
- Ensuring the security of all operations to safeguard privacy.
- Empowering data subjects with complete control over knowing who is processing their data and how, in a verifiable manner.

The Security Component provides the following mechanisms:

1. API management (including authentication, authorization, data encryption, and API logs) to ensure that data is secure, and that exposed APIs are protected;
2. Role-based access control (RBAC) to ensure that only authorised roles (and in turn RETENTION registered end-users) are allowed to access certain APIs and resources;
3. Two-factor authentication to further increase login security and protect end-user accounts from malicious attackers, prevent unauthorised access and enhance protection against phishing attacks;
4. Data encryption (for identifiers, personal information, Personal Identifiable Information - PII) resides within the Security Component repository to ensure that data is secure and cannot be accessed by unauthorised end-users;
5. API logs to monitor activity and detect any potential security threats;
6. Devices management to control the inventory of smartphone IMEIs, their status, and tracking the data transmissions (metadata only). The devices management assist the Tech Support User to promptly reach out study participants for technical assistance without compromising the identification sensitivity;

7. RETENTION Pseudonymisation. The RETENTION solution (i.e., CSB instances per pilot, GIC) was designed in such a way that the full length of the collected or produced Big data (i.e., usage data, medical data, resulting analytics and interventions) should be kept in a separated anonymised fashion, while a small subset (i.e., identifiers, personal information, PII) concerning study participants and login-profile information, to be stored with high security safeguards. Within this context, the Security Component prior to digesting or transmitting any data performs data anonymization/obfuscation transformations procedures to ensure that all data and metadata references could not make the data subject identifiable. The pseudonymisation practices are mostly aimed at replacing the pseudo patient IDs (e.g., for CSB pseudo-Id1 to pseudo-Id2 and vice versa, GIC pseudo-Id1 to pseudo-Id3 as illustrated in Figure 2).

More details about the security component and its mechanisms can be found in deliverable D6.2 “RETENTION Security & Privacy by design enabling mechanisms v1”.



**Figure 2** The CSB@SecurityComponent replaces Pseudo-Id1 to Pseudo-Id2 by which pseudonymised data record is stored in CSB@Repository and vice versa. In a similar way, CSB@SecurityComponent replaces Pseudo-Id1 to Pseudo-Id3 by which pseudonymised data record is stored in GIC@Repository





### 3.4 Global Insights Cloud (GIC) layer

The Global Insights Cloud represents a higher layer in the architecture aiming to collect information from the individual CSBs and provide relevant functionality both in terms of AI model management and in terms of analytics that provided a holistic perspective on health outcomes and RETENTION effectiveness in improving them.

#### 3.4.1 GIC Dashboard

The GIC Dashboard in its current version, presented in more detail in Appendix A.1, provides the entry point to the tools that facilitate AI model management and experimentation (see Sections 3.4.2 and 3.4.3 below). As data is accumulated from the CSB sites, it will also include policy support analytics on the basis of the relevant KPIs, comparing standard practice (KPIs for the control group) with RETENTION enhanced clinical practice (KPIs for the intervention group).

#### 3.4.2 Big Data Analytics Engine

The Big Data Analytics Engine (BDAE) involves several steps, each crucial to handling large volumes of data effectively.

The first step needs to be considering the problem that has to be addressed. Once the problem has been defined, the data that feeds the input of the algorithm has to be pre-processed and adapted to the problem's need.

In general, the component is responsible for receiving, storing, and possibly processing the data to create models. It can interact with other components like the GIC Dashboard to visualize the data, the data provision service to store data, and potentially every other component that requires access to the data. Any request to get and store the data will pass through the Security component.

Next, the data ingestion mechanism must be well defined because this is the most important step of the Big Data Analytics Engine. Considering we are using FHIR as the data storage, a pipeline that receives/sends back the data for analysis, can be created with Apache Spark, for example.

Once data will be recorded, this component will be developed and adapted accordingly during the next phase of the project.

#### 3.4.3 Model Specification Tool

The GIC instance in the architecture handles management of anonymized data, and is responsible for statistical analysis, machine learning analysis, and training of models. On the other hand, the CSB instance is designed to manage all operations related to clinical sites.

For each participating patient in the RETENTION clinical study, the data recorded from the PE instance and the clinical data recorded during hospital visits, at regular intervals, will be used to generate both statistical models as well as ML models. Based on these models, inferences can be made, patterns can be detected, and correlations can be explored, showcasing the underlying factors of the targeted diseases.

The AI algorithms will be developed using the primary and secondary endpoints criteria outlined in document D8.2 "RETENTION Clinical Trial Protocol & Evaluation Framework" and will undergo training using various patient cohorts. The predictive outcomes generated by each model will then be used for individual patient assessments.

The Model Specification Tool (MST) offers several functionalities including:



- exploratory data analysis
- data wrangling
- training machine learning models
- writing code for prediction services exposed as REST API, by facilitating model traceability through API calls to the Disease Insights model registry.

On the other hand, Disease Insight (GIC@DI) enables users to display:

- machine training experiments
- associated metadata
- artifacts
- download model binaries and artifacts,
- store machine learning parameters and outcomes such as model binary files and important ML charts.

Additionally, there are functionality requirements from D3.2 “RETENTION Architecture” that will be included in the final version of the system.

The details of the functionalities have been described in the Deliverable D5.1 “Analytics & Decision-making enabling mechanisms v1”.

The RETENTION platform utilizes several open-source tools to manage and execute its Big Data Analytics (BDA) Engine. Key components include MLFlow, MinIO, Postgres, Nginx, and Jupyter Notebooks.

MLFlow is used for managing the entire machine learning (ML) lifecycle, with tracking experiments, managing and deploying models, packaging ML code for sharing, and hosting MLflow models as REST endpoints.

MinIO is an object storage suite used in RETENTION as a replica of AWS S3 services. It allows for efficient storage and delivery of models and associated meta-data.

Postgres is an open-source relational database management system used for storing the models after the training phase.

Nginx is used to host the Jupyter notebooks that contain the data pre-processing pipeline.

Jupyter Notebook is a web-based interactive development environment used for creating Python scripts that read data from the GIC repository. These scripts utilize ML models, such as a Gradient Boosting Classifier, to create a model that responds to a specific scenario.

The machine learning process starts on the Global Insights Cloud (GIC) server where model training occurs. After training, the model is ready to make predictions on new data. The MLFlow server on the GIC helps manage and store these machine learning models and their metadata.

The Clinical Side Backend (CSB) server fetches the trained model from the GIC server via the MLFlow REST API. The clinicians can then use this model to make predictions on patient data, fetched from medical analyses and wearable devices.

Clinicians access patient data and choose models for prediction through a dashboard hosted on the CSB server. The model is called through a REST API using the patient's data stored in a FHIR database, and predictions are displayed on the dashboard for clinicians to use for decision making.





#### 3.4.4 Disease Insights

This module provides means for model understanding and visualisation of its inner structure & behaviour, providing functionality details to the Model Specification Tool. It aims to provide transparent, explainable and trustworthy insights tailored to the specific medical criteria as outlined in the primary and secondary endpoints within document D8.2 “RETENTION Clinical Trial Protocol & Evaluation Framework”. It runs via the Jupyter Notebook tool which saves into the Database along the models, the model’s metadata that provides insights (Confusion Matrix, Shap explainability, other model’s performance KPIs, as shown in A.4 “Model Specification Tool, Big Data Analytics Engine, Disease Insights”). These will be further available for visualizations within the Dashboard. Such investigation will demonstrate the validity of the model, increase trust in its provided outcome and likewise provide deeper understanding on the underlying factors that contributed to the analysed disease or specific event.

#### 3.4.5 Decision and Policy Support

Besides information coming from measurements, the platform allows entering information about the patient's whereabouts, like age or location. Also, via the questionnaire’s facility provided by the platform, patients can describe their physical as well as the psychological situation.

All these data can be used to gain insight into enforcing better policy rules for the medical profession as well as other involved parties, for example community organizations or even governmental organizations.

#### 3.4.6 FHIR and non-FHIR Data Repositories

The FHIR repository is used for the storage of medical variables such as conditions, observations, questionnaires, patient visits and hospitalisations, while the non-FHIR repository hosts environmental measures such as temperature, humidity and pollution index as well as the notifications generated by the Decision Support System. The GIC repositories will accumulate already anonymised data transferred from the CSB instances. The GIC data will be used for further analysis aiming at a continuous model refinement.

#### 3.4.7 Data and model sharing

The specialised tools for Data & Model Sharing enable third-party users to receive or bring new data or data models to the RETENTION platform. As described in D5.1 “RETENTION Analytics & Decision-making enabling mechanisms v1”, 4 specific tools are included: (i) Data sharing to RETENTION platform, (ii) Data sharing from RETENTION platform, (iii) Model sharing to RETENTION platform and (iv) Model sharing from RETENTION platform. All four tools run locally on the third-party user’s machine. These tools utilize the PYQT framework for the development of the GUI and the JSON, requests, and CSV python libraries in order to make the necessary conversions to FHIR and non-FHIR data structures of the data of interest. The necessary APIs (for example ML FLOW’s API for the models) to post and get the data and models from GIC are utilized; the operation of the tools require the integration with the security component.

The following paragraph summarises the details for each tool, which is described in detail in the corresponding deliverable D5.1 “RETENTION Analytics & Decision-making enabling mechanisms v1”.

“Data sharing to RETENTION platform” tool enables third-party users to bring new data to the RETENTION platform. More specifically, it enables the user to import a CSV in a predefined format having in place the patients and the corresponding data he/she wishes to bring to the global insights cloud. The tool checks the format of the CSV and then convert the data into the FHIR and non-FHIR structures to post them to the global DB.



“Data sharing from RETENTION platform” tool enables third-party users to receive the project’s data and use them for their own research needs. More specifically, it enables the user to receive the FHIR and non-FHIR data from the RETENTION platform and also view the online implementation guides in order to be able to decode the information that he/she receives. GET requests are going to enable the user to receive the data in JSON structures that are going to be saved locally.

“Model sharing to RETENTION platform” tool enables third-party users to create their own models based on RETENTION models specifications. More specifically, it enables the user to make adjustments in terms of pre-existing features and algorithms in a number of available datasets. Each dataset corresponds to a different use case of RETENTION, and by enabling and disabling the aforementioned variables the third-party user is able to achieve different model predictive capabilities.

“Model sharing from RETENTION platform” tool enables third-party users to acquire the trained models and use them with their own similarly structured data. More specifically, it enables the user to download the pickle files of the already trained models of RETENTION and use them on his/her own similar data. In order to download the appropriate model, the user should choose from a list of all the available clinical trial datasets (use cases). The download takes place utilizing the ML FLOW REST API.

The tools are under development. They will be reported in detail in the final version of the system. Deployment and integration activities are also in progress.

#### 3.4.8 Repository for Models

The Repository for Models is used to store the trained ML models and personalised interventions, in the form of a non-FHIR and ML Flow Repository. The transfer and download of ML models from the GIC to the CSB takes place utilizing the ML FLOW REST API.

In the RETENTION platform, the repository for models is a crucial component that manages, stores, and provides access to machine learning models trained within the platform. It's handled by MLFlow, an open-source platform designed to manage the complete machine learning lifecycle, including experimentation, reproducibility, and deployment.

Here's a more detailed breakdown of the Model Repository solution:

**Model Storage:** The models are initially created and trained in a Jupyter Notebook environment. Once trained, these models are saved to the model repository, along with their metadata, such as parameters, performance metrics, and confusion matrix or SHAP outcomes.

**Model Versioning and Lifecycle Management:** MLFlow's Model Registry allows for model versioning and lifecycle management. It provides capabilities for model stage transitions (for instance, from staging to production) with full versioning and annotation.

**Model Transfer and Accessibility:** MLFlow has a mechanism that allows for the model transfer from the repository where the model was created to the machine that calls the `load_model` function. On the CSB (Clinical Side Backend) server, the trained model from the GIC server is fetched using MLFlow's REST API.

**Model Execution:** Once the model is loaded on the CSB server, it can be used to make predictions on patient data. The Jupyter Gateway exposes these models via a RESTful API, enabling the CSB to consume these models for making predictions.



Data Security and Privacy: Access to the model repository is secured with username and password authentication, ensuring that only authorized personnel can access the models and their associated metadata.

Integration with Other Tools: All model data, including the meta-information associated with that model, is stored on an Amazon S3-like bucket with the help of Minio. This is an object storage suite that ensures compatibility and high-performance storage across various platforms.

### 3.4.9 Security Component

The security component that resides in the CSB instances per pilot and GIC is the same. Details about the security component can be found in sub-section 3.3.5.



## 4. Installation, deployment, and integration

In this chapter we outline the steps needed to perform the deployment and integration of the platform.

Further, technical documentation and user guidelines are paramount to ensure platform usability and help users navigate through its functionalities effectively. Comprehensive user manuals were created, API documentation, and other essential guides, as incorporated in Appendix. A.

### 4.1. Overall platform installation, set-up and deployment

The installation of components and configuration plan was previously created in D3.2 "RETENTION Architecture". The deployment was established to be handled by means of using Docker images and differs between system layers and some components as described in the following.

The RETENTION mobile application will be installed on project-provided Samsung Android phones by hospital IT personnel. The measurement tools (Garmin smartwatch, OMRON oximeter, pressure gauge and scale) will be paired with the application after installation, also by hospital IT personnel.

The RETENTION system is installed on the provided hardware and system software by ICCS. The operating system is Linux.

All CSB and GIC components are deployed via docker-compose, except for the nginx reverse proxy.

#### 4.1.1 GIC/CSB Dashboard

The CSB Dashboard exposes a single port for serving its front-end content (HTML, JS, CSS, images). The CSB Dashboard is parameterised so that its front-end, having the same code, will make calls to the appropriate backend, namely the front-end of the CSB Dashboard for a specific clinical site will issue requests to the backend for the same site:

1. AUTH\_URL, specifies the CSB Security component URL where user browser is redirected for authenticating the user
2. TOKEN\_URL, specifies the CSB Security Component URL for obtaining access tokens
3. MAIN\_URL, specifies the CSB Security Component URL for making backend calls (the CSB Security Component acting as a gateway to all other backend components)

```
version: "3.8"

services:
  csb-dashboard:
    image: johnzaza/csb-retention:3.2.17
    ports:
      - 127.0.0.1:8380:8042
    environment:
      MAIN_URL: "https://retention-csb-test.biomed.ntua.gr/api/"
      TOKEN_URL: "https://retention-csb-test.biomed.ntua.gr/auth/realms/retention/protocol/openid-connect/token"
      AUTH_URL: "https://retention-csb-test.biomed.ntua.gr/auth"

networks:
  default:
    name: retention-net
```



```
external: true
```

The GIC Dashboard exposes a single port for serving its front-end content (HTML, JS, CSS, images). The GIC Dashboard is parameterised, similarly to the CSB Dashboard, so that its front-end, having the same code, will make calls to the appropriate backend. While there are not many production GIC backends like there are CSB backend (one per clinical site), this parameterisation allows the same code / docker image to operate in the three different RETENTION environments: production, staging/UAT, testing.

1. `AUTH_URL`, specifies the GIC Security component URL where user browser is redirected for authenticating the user
2. `TOKEN_URL`, specifies the GIC Security Component URL for obtaining access tokens
3. `MAIN_URL`, specifies the GIC Security Component URL for making backend calls (the CSB Security Component acting as a gateway to all other backend components)

#### 4.1.2 Model Specification Tool, BDA Engine and Disease Insights

Model Specification Tool, BDA Engine and Disease Insight components are interconnected and deployed together. The installation of these three components that should take place on GIC side and parts at CSB side, is described in Appendix B.1.

On short, the installation steps include copying the components project code to the respective GIC and CSB machines, pushing two customized docker images (for Jupyter and MLflow) to public GitHub repository and starting the system using Devstack. For the CSB side, the prediction service and the connected web app is started using docker compose. Compare to GIC which contains several components such as: Model Specification Tool (MST), Big Data Analytics Engine (BDAE) and Disease Insights (DI), the CSB machine contains a simpler version with only Big Data Analytics Engine Executor (BDAEE).

#### 4.1.3 GIC/CSB FHIR and non-FHIR Data Repositories

The dockerised FHIR repository consists of two containers. The first one is built from a MariaDB image and is used for the storage of the data, while the second one is the actual FHIR server providing the functionality for the storage of the data. The MariaDB container exposes no ports hence only communicating with the FHIR server. The FHIR server exposes a different port for each clinical centre and the GIC instance. A URL parameter for the integration of the implementation guide and the insertion of the custom profiles, extensions and code systems is used.

The dockerised non-FHIR repository also consist of two containers. The first one is built from a MariaDB image and is used for the storage of the data, while the second one is the respective API which handles the corresponding HTTP methods. The MariaDB container exposes no ports hence only communicating with the API. The API exposes the same port for all instances (clinical centres and GIC) but communicates with a different instance of the database each time.

In addition, to support users, a description of the FHIR implementation guide, utilized by the technical team during development and also by the data scientist users, is presented in Appendix A.2.

#### 4.1.4 Decision Support System

The Decision Support System also runs as a docker container. It communicates with the FHIR repository from which it digests data for the generation of the notifications. Communication with the non-FHIR repository is also necessary for the storage of the notifications.

#### 4.1.5 Local Home Gateway



In order to be able to execute the above functionalities a number of installations need to take place on Raspberry Pi. The latter runs on a Raspberry OS software and the programming language that is being utilized for the necessary developments is python 3.7. One of the main services that is deployed is the open weather map service, that is responsible for gathering the external weather data. The open weather map service provides multiple APIs to be utilized in order to obtain the necessary data. First and foremost is the geocoding API that converts the known input (zip code/municipality, country initials) to a familiar, to the open weather APIs, data input, that is longitude and latitude coordinates. Then the APIs of interest can be triggered. For the RETENTION, these are the current weather data API and air pollution API. Both of them retrieve the coordinates as input and provide the EDGE computer with the necessary external weather conditions.

For fetching the sensor indoor data from the temperature and humidity sensor, the Bluetooth low energy protocol is being utilized. Two tools are installed in order to achieve the connection and data requests from the sensor, and these are bluez (<https://ubuntu.com/core/docs/bluez>) and hcitool (<https://manpages.ubuntu.com/manpages/focal/man1/hcitool.1.html>). Having these libraries in place, then it is possible to scan for nearby peripheral Bluetooth devices, achieve connection and pairing with the device, and discover the uuid characteristics of interest in order to then use the BLE protocol for the data retrieval. Automatic scripts have been prepared to run hourly and request the necessary data either from the open weather service or the sensor, convert them and store them locally and post them online through the non-fhir api. The libraries used for these tasks are NUMPY, JSON, CSV and REQUESTS as also mentioned in D6.1 “RETENTION Interfacing, Device Federation and Visualisation components v1”.

Apart from the above functionalities, the initial setup of each system is required before acquiring the necessary data and so a tool to be used by the technicians has been developed. This tool provides many options to facilitate the setup and is described in detail in D6.1. The tool is built on top of one main library that is the PyQt (<https://wiki.python.org/moin/PyQt>) framework.

#### 4.1.6 Data and Model Sharing, Decision and policy support

The Data and Model sharing, Decision and policy support tools are under development. They will be reported in detail in the final version of the system.

#### 4.1.7 Mobile App

The application will be downloaded and installed by hospital IT personnel on the Samsung A03 Android devices that will be provided to the patients as part of the project. Alternatively, if a patient wishes to use their own phone, they can download the app using the following link: <https://play.google.com/store/apps/details?id=com.datamed.retention&pli=1>

Note: The application is only compatible with Android versions 12 or higher.

Instructions on how to install the application, as well as how to pair the measurement devices with it, are available as part of the RETENTION app manual in Appendix A.5. <https://play.google.com/store/apps/details?id=com.datamed.retention&pli=1>

## 4.2. Integration

### 4.2.1 Platform integration

The platform was presented in depth in D3.2 “RETENTION Architecture”, and a short overview will be given in what follows.



The platform comprises the following logical sub-components, within the:

- GIC Layer: Dashboard (\*), Model Specification Tool (\*), Disease insights (\*), Big Data Analytics (BDA) Engine (\*), Decision & Policy Support (\*), Data models repository (\*), DB-data (FHIR and non-FHIR), Security component,
- and the CSB Layer: BDA Engine Executor (BDAEE), DB-data (FHIR and non-FHIR), Data trained models (\*), Decision Support System (DSS), Dashboard, Security component.

The components marked with (\*) will be implemented in version 2 of the platform.

For the Software-Hardware integration of the RETENTION Platform, a staged approach is used, as described in D3.2 “RETENTION Architecture”, consisting of two stages: Stage 1: Installation and Stage 2: Configuration and Adjustments. Once the stages are completed, the system is configured and ready for testing by users.

### **Stage 1: Installation**

The installation comprises the following steps:

1. Virtual or On-site inspection of the infrastructure that will be used
2. Study of installation requirements of the individual applications and scheduling of actions
3. Creation and preparation of virtual machines
4. Creation and generation of docker containers from each component owner
5. Gather the required docker containers for the installation
6. Installation of the software stack in the available infrastructure:

#### Integration of the system

- a. assembling all the containers provided by different component owners to ensure there's a unified deployment process.
  - b. setting up the necessary software, libraries, and dependencies required for the platform to function.
  - c. ensuring the various modules, databases, and systems in both the GIC and CSB layers interact with each other seamlessly.
7. Customisation of software for integration in the network environment: tweaking and adjusting software components to fit into the specific network or IT environment, ensuring things like IP addresses, domain names, and ports are correctly configured.

At the end of the phase, the initial environment of the above applications will be configured for every module (GIC, CSB, EDGE, and the Mobile Application).





## **Stage 2: Configuration and Adjustments**

At this stage, the necessary adjustments and configurations of the Applications will be made, in relation to the requirements of the individual services of the RETENTION System. The system and the application software customisation teams will deal with the configuration and adjustments that the system needs to function correctly as described in this document and the requirements. The Application Customisation and Design will consider the operational procedures of the project, the needs of the clinical team and needs of the system, and they will be mainly concern:

- User groups and the rights that will be assigned to them
- Roles they will play during the operation of the system
- Particularities of the clinical teams
- The offered infrastructures
- Objectives of the project (as declared in the DoA)
- Organisation and Administration of the project, and
- Other elements that emerge during development, stage 1

### **4.2.2 Integration technology**

The system integration uses Docker technology (<https://www.docker.com/>). Each of the logical subcomponent is contained in Docker containers, cumulating a total of 65 containers.

### **4.2.3 Overall system integration**

In this section, we will describe the system integration, meaning the interconnection between the components of the platform. Integration provides organizations with the necessary tools to connect their systems, applications, and data across their environment. With continuous integration, the code resides in a repository and every time new code is pushed to the repository, it can be verified by a set of automated unit and integration tests, after which an automatic new build is triggered and is set to be deployed. This pipeline prevents the deployment of faulty implementations and exposes bugs at compile time rather than runtime. To this end, the RETENTION application uses Docker tools for deployment of said components.

Docker is an open platform for developing, shipping, and running applications. It also enables separation of the application from infrastructure, allowing quick delivery of the software. Containers can be extremely lightweight and contain everything needed to run the application, without any requirements of external dependencies. As a result, they can be easily shared and deployed without worrying about the differences in the underlying infrastructure. If it works on your Docker setup, it can work on any Docker setup.

The general idea of RETENTION platform is the use of microservices architecture. The application works as a collection of services that are independently deployable, loosely coupled, organized around business capabilities, owned by a small team and are highly maintainable and testable. The microservice architecture enables the rapid, frequent, and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

For these reasons, Docker is an excellent tool for managing and deploying microservices and it is used in RETENTION application platform. Each microservice can be further broken down into processes running in separate Docker containers, which can be specified with Dockerfiles and Docker Compose configuration files.





Individual component owners are responsible for supplying a valid image to an image registry like Docker Hub or Gitlab's registry. From there, images are pulled and containerized through a separate docker-compose.yaml file for each hospital instance.

All instances are stored in the production environment in /retention/deployment. Within this depository, is a separate folder for each hospital that contains a combination of said docker-compose.yaml file and a .env file which contains usernames, passwords and other data that can be used to parametrize the corresponding docker-compose.yaml file.

Every hospital instance's backend can be brought online or offline using standard docker commands. For convenience, the technical team has developed shell scripts in the /deployment folder that allow the administrator to perform these actions for all hospital instances.

Separately, in a folder called /dashboard a different docker-compose.yaml file is stored, that is responsible for the declaration of all instances' dashboards. Using this setup, the dashboard component owners can quickly change the version of the dashboard and redeploy.

Every instance requires a unique docker network. This can be performed by 'docker network create <hospital>'. Once that network is defined externally, it should be declared in the docker-compose.yaml file of the instance.

Once an instance is brought online, it is initialized and all components begin operating in a specific order, as some of them rely on the health of others. Once initialization is complete, local volumes defined for the components are created within the /volumes folder in the folder of each instance. Additionally, for data security purposes, we have a separate container that creates database backups. Those files are stored in the /backup folder.

#### **4.2.3.1 Patient Edge**

Patient Edge components (RETENTION Mobile App and Home Gateway) as well as the CSB Dashboard front-end communicate with the CSB Security Component via REST requests (e.g., for obtaining access tokens) and with CSB Backend (FHIR/Non-FHIR Repositories) via REST requests that go through (if authorised) the CSB Security Component. The same applies for the GIC Dashboard front-end, the GIC Security Component and the GIC backend services.

#### **4.2.3.2 Data and Model Sharing**

The Model trained on GIB is shared on CSB side by following the three steps:

1. Annotate the notebook with the HTTP method and the API path (e.g., # POST /heartf/predict). This makes the API accessible via a URL (e.g., http://csb\_host\_machine:host\_port/heartf/predict) using the HTTP POST method.
2. Once the Data Scientist has completed the prediction notebook, a DevOps engineer will:
  - a. Securely copy this notebook to a specified folder on the CSB side (<csb\_installation\_folder>/retention/jupyter/notebook/retention-api).
  - b. If a new notebook is added, update the docker-compose.yml file to include a new service definition.
3. Restart the stack using docker-compose restart.

In Version 2 of the platform these steps will be fully automated.



The saving and loading of the models are performed using MLFlow Model Registry (<https://mlflow.org/docs/latest/model-registry.html>) and Minio (<https://min.io/>).

#### 4.2.3.3 Local Home Gateway

As far as the integration with the rest of the RETENTION platform is concerned, it is realised through the final stage of the local home gateway's functionality which is the transfer of the converted non-FHIR JSON structures to the GIC DB. All the other functionalities of the system work separately and independently from the rest of the platform, such as the fetching of the raw data and the local storage of them. The first level of integration is being done through the security component. This happens through the token-based authentication that takes place by providing the correct credentials to a specific GIC token URL. After the completion of this process an access token is being sent back to the local home gateway to be used for the transmission of data. The next level of integration refers to the successful posting of the non-FHIR data to the non-FHIR database. Specifically, here the integration happens through the received token and also through a post URL, that refers to the Clinical Site Backend (CSB) of interest. The final step is the necessary ids to be included in the data structure in order for the security component to be able to retrieve them successfully. These are the user Id (pseudo anonymised patient) and the package Id (a unique serial number referring to the package of the devices). The data then are stored successfully to the GIC DB, and the integration between the local home gateway and the rest of the platform (through the non-FHIR GIC DB) is achieved.

#### 4.2.3.4 Security Component

During the early phase of the project, the Security Component's source code has been privately hosted on Sphynx's Git instance. Specifically, the following repositories were created for each component:

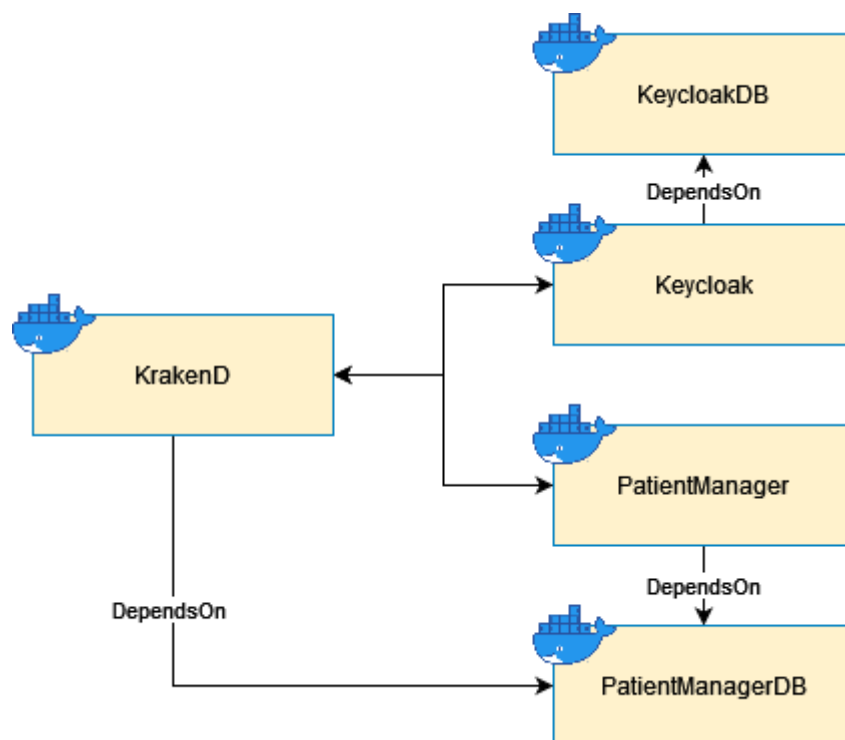
1. KrakenD: This repository holds the Dockerfile of the API Gateway component, as well as the source code of the pseudoanonymization plugins and any other necessary configuration to deploy the component.
2. Keycloak: The Identity Management component has been customized to fit the RETENTION project. This customization includes a visual overhaul of the login page and modification of the text of email notifications.
3. PatientManager: This repository contains the full source code of the patient manager component, along with its Dockerfile.
4. Deployment: This repository contains the docker-compose file and its associated environment variables required to deploy the previous components to the server. Cloning the deployment and running the docker-compose file automatically downloads and runs all necessary container images, eliminating the need to download the complete source code of the previous three repositories.

Each of these repositories had a CI/CD pipeline set up to automatically build container images when changes were made to the "Main" branch of each component, ensuring that the images were always up to date with the latest stable changes.

To download the container images from Sphynx's private Git repository successfully, authentication with the Container Registry is required. A Group access token was created, granting "Read" access to the registry's containers. To authenticate, run the docker login command:

```
docker login registry.example.com -u <username> -p <token>
```

Once authenticated, the `docker compose pull` command retrieves the version of the images specified in the docker-compose file from the private container registry. As shown in Figure 3, KrakenD interacts with both Keycloak and PatientManager. KrakenD uses Keycloak to verify that users have been authenticated and have received a valid token, and also acts as the Gateway to PatientManager. Similar connections would be depicted between KrakenD and all the backend services that it protects. Additionally, KrakenD depends on the PatientManagerDB as it directly queries the database to perform pseudoanonymization by substituting patient IDs with their pseudo-IDs and vice versa. Finally, Keycloak and PatientManager depend on their respective databases, KeycloakDB and PatientManagerDB.



**Figure 3. The containers that comprise the Security Components, their connections, and their dependencies.**



## 5. Technical Testing & Quality Assurance

This chapter provides the definitions of the technical testing and quality assurance, as well as how these were implemented in RETENTION project.

As a software product progresses through its development cycle, identifying and resolving issues becomes increasingly challenging and costly, especially as it nears its release date. Therefore, it's crucial to assess the software's quality in both design and functional aspects. This evaluation involves various tasks, such as developing test scenarios to measure system performance, inputting sample data, and customizing special features.

By defining the requirements and standards, in deliverable D3.1 “RETENTION requirements”, we cleared the functional and non-functional requirements, as well as the performance, security, and usability standards, that the platform must meet. This will serve as a baseline for evaluating the platform's quality.

In the current deliverable we present the testing efforts that we have made to assure that the platform in its initial form is ready to be used by clinicians for the main purposes of patients creating and ingesting patients' data from various sources.

Future efforts to assure the quality of the platform will be followed in accordance with Chapter 6: “Next work and strategy”.

### 5.1 Technical Testing

After the deployment of the components, several testing mechanisms were discussed in D3.2 “RETENTION Architecture”. To mention a few, there are automated unit tests that each partner has the responsibility to implement for their corresponding developed modules, integration tests, user acceptance tests that will be conducted by the clinical teams, or integration tests.

In addition, prior to rolling out the 1<sup>st</sup> version of the RETENTION platform to the pilots, represented by the instance of Clinical Site Backend (CSB) and Patient Edge (PE), a virtual Pilot of Pilots (vPoP) process was established to fine-tune and improve the quality of the system. It involves a virtual pilot (in terms of “end-users” type) and smaller pilot (in terms of number of “end-users”) in a controlled environment, and using the same equipment later to be handed over to study participants, prior to the start of the Clinical Trial, to obtain real life data (of “test patients”) and guide the system design before it is fully deployed. It aims to replicate the conditions of the main pilots, allowing the team to identify and troubleshoot the potential issues of the platform's functionality and usability, and timely fix them. During this process, RETENTION technical partners acted with different roles supported by the RETENTION platform, such as Clinical Case Managers and Tech Support, and utilised both the equipment (home devices/sensors), Mobile App and the primary Clinical Site Backend (CSB) e-services, to simulate real-world use. VPoP offered a demonstrable view of the RETENTION solution layers (CSB, PE), and gathered valuable early expert evaluation feedback to be presented in forthcoming D8.4 “RETENTION Clinical & Technical Assessment & Validation Report v1”. With respect to the GIC layer, parts were developed to support the consolidation of different GIC components, however, the primary focus was the CSB and PE layers.

In parallel, along with the planned development and integration activities, an applicable testing plan (including integration testing, 2nd - 3rd level technical support) was also formed and applied, within the context of the vPoP, as described in the following sections.



### 5.1.1. Testing at the level of each module

Testing focuses on individual components or modules of a software application. Each partner was responsible for setting-up and performing unit testing on their components to ensure that each module or function works as expected, validating that the code is correct, reliable, and efficient. Using this technique within the platform, we will be able to detect bugs during the development process, making it easier and less expensive to fix them. This is an ongoing effort that has so far focused on the key units to be tested in each case and coverage of unit testing is expected to widen in the forthcoming months ensuring that any changes can be made without adverse unexpected effects, as these will be caught at a very early stage.

#### **RETENTION app testing:**

1. Internal testing: The application and related equipment was internally tested by members of the development team.
  - 1.1. All Garmin and OMRON devices were tested thoroughly by several members of the team, both regarding their main operation (taking measurements) and integration with the application (connectivity, result parity etc).
  - 1.2. The application itself was tested extensively, both during development as well as later during the writing of the manual.
2. Technical team testing: The technical team tested the application, providing feedback through comments to the technical team. As a result of said feedback, the application's logic changed to using wizards.
3. Clinicians testing: Clinicians testing was done using the Test staging environment. Feedback was provided in video form and was acted upon by the developers.

### 5.1.2. Integration testing

Integration testing involves testing how different modules or components of a software application interact with each other. This type of testing is performed after unit testing and helps identify issues related to the interfaces between modules, such as data flow, communication, and coordination. Integration testing helps ensure that the system works as a whole and that the individual components fit together correctly.

#### **CSB Dashboard**

The importance of providing a high-quality user interface to the RETENTION CSB is highlighted by the extensive testing performed by the partner responsible for delivering it (AEGIS), by the testing task leader (SIE), by the technical partners in unison in the context of the vPoP and by the testing performed by clinicians. In addition to these tests, repeated as needed (e.g., when new features are added or changes and improvements are made in response to end-user requests), the testing task leader also created a testing suite based on Selenium which automates the testing of a gradually expanding number of use cases.

#### **GIC Dashboard**

Due to the fact that the core functionality of the GIC Dashboard is being provided by well tested established data analysis tools, only user-based testing was deemed necessary at this stage.

#### **Mobile App**

Integration testing was performed during each phase of application testing, ensuring that test data was properly transmitted to the CSB.



Integration testing was also performed inhouse regarding the interfacing of the application with the various Garmin and OMRON devices that will be used to record patient data.

Beside manual testing, initial scripts for automated testing were written in Python to test each component individually.

### **Local Home Gateway**

Regarding the local home gateway, integration testing was held for each new updated version of the system. Specifically, the first part of integration testing made sure that the communication of all the necessary components that are responsible for gathering all the necessary data, was successful. These components consist of the temperature and humidity sensor, the open weather API, as well as the software that runs on raspberry pi and handles the automatic data gathering. The test was considered successful when all the data was automatically gathered in the raspberry pi's local database.

The next part of integration testing checked that the local home gateway connected successfully with the CSB database through the security component. This was achieved by acquiring the required token from the security component through a validation process, and then using this token in order to transfer the non fhir data to CSB database via the non-FHIR api. The test was considered successful when the data was visible in the CSB database or in the dashboard after the aforementioned process.

Finally, in order to facilitate both the local home gateway's functionality as well as the integration testing, all the functionalities were broken down in individual scripts. Each one of the scripts was executed successfully in each integration testing that was held, as well as the interconnection of these scripts that comprises the whole functionality of local home gateway. This interconnection is achieved through a UI tool available to the technician for the installation of the system. Taking the above into consideration the Home Gateway has been extensively tested in order to ensure both UI aspects and interfacing with sensors and external APIs function as expected.

### **5.1.3. Use case testing, end-to-end testing**

Use case testing focuses on testing the functionality of a software application based on real-world scenarios or user stories. Use case testing helps verify that the application meets the requirements and expectations of its end-users and ensures that it provides a satisfactory user experience.

Our team conducted use case and end-to-end testing. The tests targeted the key features of RETENTION modules and ensured the stability, accuracy, and reliability of our code with correct outcomes.

Momentarily, the testing endeavours for functionality have been manually performed by each partner to the level of their module and integration and complete end to end testing have been manually performed by the technical testing leader team (T7.2 "Technical Testing & Quality Assurance of RETENTION Platform") on a regular basis and also after any components changes. Automated tests are in development and will be incorporated in platform v2.

The vPoP involved about 100 virtual test patients on testing and staging environments, including control and monitoring groups and all three categories Heart Transplant (HT), Heart Failure (HF) and Left Ventricular Assisted Devices (LVAD). Test user accounts were also created for clinical partners to use and test the CSB (VPoP) functionality, offered as clinical case managers (7 users) and some as Tech Support (2 users).

Any problems identified were communicated to the partners responsible for the integration and testing of the platform (WP7 "Platform Integration & Testing"), and the partners responsible for the development of



each platform module. Basecamp Kanban board was used for issues management and tracking, and any potential issues discovered were immediately addressed.

### 5.1.3.1. End-user Scenarios Prerequisites

To complete the necessities for implementing the VPoP, the dashboard URL, admin account and end-users accounts are needed. Also, patient data, both medical as well as personal data is required for each scenario (see an example in the table below).

**Table 2: End-user scenarios prerequisites: set-up and data.**

<b>CSB(VPoP) Dashboard url</b>	http://147.102.33.191/auth/login			
<b>End-users accounts</b>	<b>Role</b>	Patient	CCM	Admin
	<b>Name</b>	A	John Smith	John Doe
<b>Patient Personal data</b>	<b>Pilot Organisation</b>	VPoP		
	<b>Full name</b>	Roger Schmidt		
	<b>Birthday</b>	1/1/1950		
	<b>Sex</b>	male		
	<b>Home address</b>	Harry-Blum-Platz 2 Cologne, Germany (irrelevant), PO: 50678 (irrelevant)		
	<b>Phone</b>	(+49) 02213900 (irrelevant)		
	<b>Email</b>	u501retention@gmail.com		
	<b>Recruitment status</b>	accepted		
	<b>Date of participant's consent form (dd/mm/yyyy)</b>	02/11/2022		
	<b>Withdraw consent date</b>	{null}		
	<b>Responsible clinician</b>	John Smith		
<b>Caregivers/relatives (emails)</b>	Example@gmail.com			
<b>Medical data</b>	<b>Baseline history</b>	params to be filled based on definitions		
	<b>Medication</b>	params to be filled based on definitions		
	<b>Observations (to pre- exist -personalized descriptive analytics</b>	Systolic - Diastolic blood pressure (at least 5 daily measurements each)		





	to display the following)	Weight (at least 5 daily measurements each)		
		Oxygen saturation (at least 5 daily measurements)		
		Home humidity (at least 5 daily measurements)		
		Temperature (at least 5 daily measurements)		
<b>Devices</b>	Smartphone A associated to HF Patient A (preconfigured)	Pressure meter		
		Weight scale		
		Oxygen saturation meter		
		RETENTION App installed		
	Raspberry Pi associated to HF Patient A (preconfigured)	Home humidity & temperature sensors		
		Xiaomi MI monitor temperature & humidity		

<http://147.102.33.191/auth/login>[http://147.102.33.191/auth/loginmailto:myexwife@gmail.com](mailto:myexwife@gmail.com)

### 5.1.3.2. Roles and accounts

There are several different roles like Clinical Case Managers, Tech Support, or Data Scientists within different organizations involved in the (Virtual Pilot of Pilots) VPoP. Each role is associated with email, username and other identifying parameters.

### 5.1.3.3. Flows

The testing procedure was as follows:

- Each partner performs installations on their own devices set: Set-up Home Gateway, install Mobile App
- End-user test accounts related to the VPoP were manually created by the Sys admin, to be used by all (Admin, Clinical Case Manager, Tech Support)
- Log-in into the CSB Dashboard
- Create test patient and details and verify presented information
- Assign devices IDs (smartphone and RASP-PI)
- Record data using devices and sensors
- Verify the functionality and usability of each section of the mobile app
- Verify the functionality and usability of each section of the Home Gateway app
- Verify if data was sent correctly to the respective FHIR or non-FHIR repository
- Verify if data reaches the CSB Dashboard and is correctly presented.
- Add clinical data via the respective Dashboard forms, as a clinical user type
- Verify if the clinical data is correctly stored in the FHIR repository and if is correctly presented in the Dashboard





- Check each section of the Dashboard

The tests were performed manually on each component separately to ensure that each component functions correctly, the following flows for each component were done:

Web Application (Dashboard):

1. Tested all the functionalities in the dashboard, from the login and patient creation, to adding different types of data for the patients that were created.

Mobile Application:

1. Set up the application and assess the overall ease of use
2. Paired the testing devices with the mobile application
3. Manually tested each feature of the application to check if it works accordingly
4. After the testing on the mobile application was completed, the Dashboard was checked to see if the requests that were made from the app were successful in putting the collected data inside the FHIR database, from where the Dashboard could interact with them.

Home Gateway, Raspberry PI:

1. Setting up the RPI according to the user manual provided by FORTH, alongside with the provided ISO image for the already set up operating system.
2. After the RPI was set up the application that runs on the raspberry pi was set up, and the IoT sensor was connected through a Bluetooth connection to the RPI.
3. Manually tested posting data from the RPI sensor to the non-FHIR database.
4. Checking the data made it to the database, and also the Dashboard was checked to see if the appropriate data was displayed.

FHIR/non-FHIR Data Repository:

1. Verifying if the server works properly, checking the API calls if return correct information
2. Verifying if data is saved in correct form following the defined structure

ML Application (Dashboard, BDAE):

1. Verifying the correct ML problem definition, the correct implementation, and the usability regarding the presented model results

In this version, for examples, flows like Create Patient -> Check FHIR repository -> Add Device Data -> Add Medication were used to check the health of the platform.

#### **5.1.3.4. Reports**

After each and every testing session that was performed, reports were made with bugs that were detected for each component as well aspects that can be improved from the usability point of view. The recurrence of end-to-end testing and reporting varied from monthly to more frequent occurrence up to few days, in response to development changes. The reports were uploaded on Basecamp, shared with the entire consortium.

An overview of the reports that were done can be seen below:

Web Application (Dashboard)



1. Hospitalizations and Medications: Restrictions are needed on date fields and dosages. For example, dates for hospitalizations should not be allowed beyond the next 5 years, and medication dosages need user limits.
2. ECG and Echocardiography Forms: User indicators/suggestions should be added for PQ/PR, QRS, and QT markers.
3. Forms and Dashboard:
  - Forms should not be submitted if only the date is provided.
  - Dashboard data could be better centred.
4. Basic Patient Info: Should have input validation to prevent out-of-bounds values like height being 2000 cm.
5. Inactivity Timeout: The application should automatically log out users who are inactive for too long, instead of just showing an error.

#### Mobile Application

1. Device Pairing Issues: Various issues ranging from foreign devices wrongfully paired to other pairing malfunctions.
2. Questionnaire: Different typos, missing questionnaires and minor connectivity errors
3. Device Issues: Mainly submission errors in relationship to the CSB dashboard
4. Miscellaneous: Security, aesthetics and user experience issues
5. New JSON Structure: Regular changes to the JSON structure, while adding required variables over time

#### 5.1.4. UI testing

For the UI testing there are two ways to realise this:

1. Manual UI Testing: this is done each time a component is updated, and an operator executes the basic flows. These tests are already performed recurrently
2. Automated Testing. Using tools like Selenium will be implemented further, including flows like the one above to test the Dashboard and the Mobile Application.

The UI interaction has been manually tested in this phase and as more developments will take place, the tests will contain automated testing procedures.

#### 5.1.5. Security testing

Security testing is performed to identify vulnerabilities, threats, and risks in a software application and to ensure that the system and its data are protected from unauthorized access, data leakage, or other malicious activities. Security testing includes various techniques, such as penetration testing, vulnerability scanning, to identify potential security issues and evaluate the effectiveness of security controls in place.

Security testing procedures including penetration testing and the security measures for safeguarding patient data as part of the continuous Security and Privacy Assurance Platform (cSPAP) component have been presented in D6.2 "RETENTION Security & Privacy by design enabling mechanisms v1". As the RETENTION platform progresses, aspects regarding the security testing will be updated in D6.4 "RETENTION Security & Privacy by design enabling mechanisms v2".

For the validation of the security component and its integration to the RETENTION platform, unit tests were used. Specifically, for the Patient Manager component, comprehensive unit testing was employed to validate

the functionality and correctness of the PatientService class. Unit testing is a critical component of the development process, allowing the identification of issues early in the development lifecycle.

The unit tests were conducted using the JUnit 5 testing framework and the Mockito library. Mockito is utilized for mocking the repository dependencies (e.g., Patient Repository, Organisations Repository, and Fhir Repository) to isolate the Patient Service class from the database interactions. This allows testing the service's logic independently and ensuring that each unit operates as expected.

The test cases covered the creation, updating, and deletion of a patient, including the assignment of caregivers and clinicians to them. Sample code from one of the unit tests of the security components is shown in the following figure.

```
package ch.sphynx.patientmanager.service;

import ch.sphynx.patientmanager.dto.*;
import ch.sphynx.patientmanager.model.Caregiver;
import ch.sphynx.patientmanager.model.Organisations;
import ch.sphynx.patientmanager.model.Patient;
import ch.sphynx.patientmanager.model.Pilots;
import ch.sphynx.patientmanager.repository.FhirRepository;
import ch.sphynx.patientmanager.repository.OrganisationsRepository;
import ch.sphynx.patientmanager.repository.PatientRepository;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;

import java.time.Instant;
import java.util.*;

import static org.mockito.ArgumentMatchers.any;

@ExtendWith(MockitoExtension.class)
public class PatientServiceTest {
    @Mock
    PatientRepository patientRepository;

    @Mock
    OrganisationsRepository organisationsRepository;

    @Mock
    FhirRepository fhirRepository;

    @Test
    public void testCreatePatient() throws Exception {
        PatientCreateDTO p = new PatientCreateDTO(
            "John Doe",
            "01/01/1970",
            "1234 Main Street",
            "12345",
            "123-456-7890",
            "johndoe@retention-project.eu",
            1,
            Date.from(Instant.parse("2021-01-01T00:00:00.002")),
            1,
            Set.of(
                new CaregiverUpdateDTO("caregiver1@retention-project.eu"),
                new CaregiverUpdateDTO("caregiver2@retention-project.eu")
            )
        );
    }
}
```

Figure 4: Sample of unit test from the security component

### 5.1.6 Performance testing

Stress test was used to test the system's performance under extreme conditions. The capability of FHIR repository and the Dashboard visualizations to handle vast amount of data at once were tested by mimicking an overload of data. Particularly, 16K patients, 12K conditions and 10M were transmitted to the FHIR database divided to bundles of 500 resources (maximum resource capacity of a FHIR bundle). The FHIR repository responded positively to this test, which largely exceeds the needs of the RETENTION project. The CSB dashboard visualizations were affected, but fixes have been followed to support the case.

### 5.2 Quality Assurance



Following standard practice in development to ensure platform quality, testing and staging environments were created, following the same implementation and architecture. The testing environment ensures each application component functions correctly and the staging environment ensures the components still perform well when deployed in a live environment.

Beyond test usage, recurrent demo sessions were organized to receive feedback from the clinical team. The clinical team provided insight into each endpoint by thoroughly describing their needs and functionalities. In the event of having further questions, we communicated via email, Basecamp, online meetings and phone calls. Furthermore, the clinicians offered their input for the technical side, regarding the endpoints, thus we were able to easily connect our findings and issues with partners through the aforementioned reports and meetings. In this way, the feedback sessions were essential for the overall quality and development of the platform. Physicians provided feedback in video form or feedback forms after assessing the RETENTION application in the Test/staging environment. The feedback was collected by the technical team and appropriate actions were taken after assessing it.

As examples of the actions followed during the demos, are presented in the table below:

**Table 3: Example of VPoP demo testing actions for the functional assessment.**

Application	Function
CSB	Registration and Login (all roles)
CSB	View all patients
CSB	Search for a particular patient
CSB	Create a new patient
CSB	Personal data/Overview/Demo etc
CSB	Pair a device
CSB	Insert medical data
Mobile App	Using Pressure meter
Mobile App	Using Smart scale
Mobile App	Using pulse oximeter
Home Gateway	Using Home humidity sensor (passively)
Home Gateway	Using Temperature sensor (passively)
CSB	View HF Patient B data
CSB	View full medical record
CSB	View existing observations
ML	Survival Rate in 250 days Prediction (test use-case)
ML	Heart Failure Risk Prediction (test use-case)



As examples of scenarios followed during the demos, are presented below:

#### **Scenario A:**

Patient Data Viewing: Involves logging in as a Clinical Case Manager (CCM) to view all patient data.

Data Collection: Describes the collection of various measurements (e.g., blood pressure, weight, etc.) using different devices, both on a mobile app and a Raspberry Pi.

Data Analytics and ML: Incorporates machine learning models to predict survival rates, which links to the database where patient data is stored.

Example of a Possible 2nd Scenario: Describes a Heart Failure Risk Prediction based on several medical parameters like Chest Pain Type, Resting Blood Pressure, etc.

#### **Scenario B:**

Dashboard Usage: Discusses the Clinical Case Manager logging into the dashboard, viewing all patients, and potentially making a typo during the login.

Patient Creation: Details the steps to create a new patient profile in the system.

In the next platform version, more elaborated testing scenarios will be addressed and more various user roles.

### **5.2.1 Robust development process**

To assure that the development process is robust, we have adopted an Agile software development process, to ensure that the platform is developed in a controlled, systematic, and efficient manner. This helps maintain the quality of the platform throughout the development process.

The Basecamp platform helps implementing this method by offering tools like Kanban Board.

### **5.2.2 Collect and incorporate user feedback**

The procedure for collecting user feedback was realized in two ways:

- Using surveys forms with System Usability Scale (SUS) to gather direct feedback from clinical users about their experience during demos and their personal use of the platform.
- Video form, capturing the actions performed on the mobile App, for example.



## 6. Next work and strategy

Moving forward to the 2<sup>nd</sup> platform version, we plan to intensify our efforts on development, testing and integration fronts, further described in this section.

### 6.1 Further development

Considering development, the focus will be on the GIC layer side, targeting data modelling with the cumulated GIC data repository, GIC Dashboard improvements, ML models implementations, Data and model sharing, Policy Support, and overall feature updates.

Further development on the RETENTION application:

- i) Update check functionality to be implemented. The application will check for updates automatically and block usage if the latest version isn't installed, while also prompting the user to update it.
- ii) Other features will be defined once we retrieve more feedback from clinicians

### 6.2. Testing planning

For our next steps regarding testing and integration, we plan to develop more automated testing tools for both functional and non-functional requirements. This could include unit tests, integration tests, load tests, security tests and more to cover a wider array of test cases and edge cases, ensuring that our platform performs optimally even under unexpected conditions. The integration tests will verify the validity of the general flows like creating a patient, adding measurements for that patient and checking if the values are correctly displayed on the dashboard. All these tests will be implemented with mostly Python scripts or use of libraries like Selenium that will be run on demand when new features for the overall platform components appear.

After each significant update, comprehensive integration tests will be conducted to validate the seamless operation of all components. Every new feature or significant change will be accompanied by relevant documentation updates. This will help to maintain a clear, updated reference for the development teams. Feedback from users and experts will be utilized to improve these resources continually.

All new features will be developed in the Test and Staging environments before moving on to Production (Hospitals) to ensure that most of the bugs that could cause unpleasant issues are caught early and are not transferred into Production.

For planning the 2<sup>nd</sup> platform version updates and development, we consider gathering user feedback on the current platform, analyse the performance of the existing platform, and identify areas that require improvement or new features that can be introduced. Then we will prioritize these features/improvements based on their impact, feasibility, and alignment with our overall strategic objectives.



### 6.3. Quality Assurance planning

Quality Assurance (QA) planning involves developing a strategy that the platform will function properly under expected conditions and given requirements. This is a process that ensures that a software platform meets the specified requirements and delivers a high level of performance, reliability, and user satisfaction. To achieve this goal, peer reviews and dedicated QA cycles are planned for each release, focusing on functionality, performance, security, and usability to ensure high-quality platform. Four steps will be followed as described in the subsections below.

QA will be performed separately on each module of the system by the appropriate partner, and the data will be collected and analysed to see the overall quality of the hole platform.

#### 6.3.1 Elaborate the testing procedures.

Perform more elaborated testing procedures like system, performance, coverage, mutation testing. For this we will use automated testing tools and frameworks to improve the efficiency and coverage of your testing efforts.

#### 6.3.2 Continuously collect and incorporate user feedback

Further improving the user feedback collection, we will focus on using online surveys to gather direct feedback from users about their experience, using a mix of open-ended and closed-ended questions to gather both quantitatively as well as qualitative data. We can extend the feedback not only for platform users, but also on 3<sup>rd</sup> party users, as with the development of the Data and Model Sharing tool to 3<sup>rd</sup> party users.

#### 6.2.3 Continuous improvement of the platform

Use the insights gained from testing, monitoring, and user feedback to continuously refine the platform and the quality assurance process itself. This will help ensure that the platform remains high-quality and up to date with evolving user needs and industry standards.



## 7. Conclusions

In this document, we have detailed the objectives, strategies, and progress of the RETENTION project. We have showcased how this initiative combines the expertise of several different organizations and employs an array of technologies and tools to work towards a common goal.

Beginning with the architecture of our platform, we have described the various components, their functionalities, and the critical role each plays in the broader system. Each of these components, from Patient Edge the GIC layer, has been developed and implemented with the goal of enhancing the capabilities and efficiencies of the platform.

We have outlined the process of installation, integration, and deployment for each component, showcasing the interconnectedness of our operations and the seamless functionality of the overall platform. In addition, we have discussed the importance of usage guidelines to ensure that all components are utilized effectively and optimally.

We have highlighted the importance of rigorous testing and quality assurance in ensuring that our platform performs optimally under various conditions. Our goal is to deliver a high level of performance, reliability, and user satisfaction.

Looking ahead, we have discussed our strategies for the next phase of our work, focusing on the critical aspects of testing and quality assurance planning. Our approach will ensure that we continue to refine our platform, enhance its functionality, and increase its reliability and performance.

In conclusion, the RETENTION project reflects a comprehensive, multi-disciplinary approach to problem-solving. It underscores our commitment to leveraging cutting-edge technologies and innovative methodologies to create a platform that meets the needs of our users and delivers value to all stakeholders. As we move forward, we are confident that our concerted efforts and strategic planning will lead to the successful execution of our project objectives and contribute significantly to the advancement of our shared goals.





## Appendix A. Usage guidelines

Preliminary Documentation and Guidelines for usage of RETENTION platform relates to first phase of Task 7.3 “Documentation & Guidelines for usage of RETENTION platform” leading to the preparation of the end-user’s manual (guidelines and end-users flows for CSB, PE (including mobile, RPI, sensors)). The manual will be a web-based service including FAQs and videos to be used by end-users (clinicians and patients) to understand how the RETENTION solution works and how to fix configuration issues autonomously.

### A.1. CSB Dashboard and GIC Dashboard

The user guides for the CSB Dashboard are available on RETENTION’s Basecamp repository, and also provided as a supplementary material along with this document:

User Guide for Clinicians (S1\_CSB Dashboard - User Guide for Clinicians):  
<https://3.basecamp.com/4281217/buckets/22373922/uploads/6503355711>

User Guide for Technicians (S2\_CSB Dashboard - User Guide for Technicians):  
<https://3.basecamp.com/4281217/buckets/22373922/uploads/6503355795>

The GIC Dashboard is presented in D7.1 “Integrated RETENTION platform v1”. In its current version, in alignment with the fact that the RETENTION clinical trial has not yet commenced, it currently serves the needs of Data Analysts in the project whose work is to serve by Jupiter Notebook and ML Flow. Both platforms have extensive documentation online. Our recommended guides are:

MLFlow: <https://www.mlflow.org/docs/latest/tutorials-and-examples/tutorial.html>

Jupyter Notebook: <https://docs.jupyter.org/en/latest/> <https://www.mlflow.org/docs/latest/tutorials-and-examples/tutorial.html>  
<https://www.mlflow.org/docs/latest/tutorials-and-examples/tutorial.html>  
<https://www.mlflow.org/docs/latest/tutorials-and-examples/tutorial.html>

### A.2. FHIR Implementation guide

RETENTION implementation guide is a web portal describing the FHIR Implementation Guide for the RETENTION Project. Details on FHIR Resource Profiles, Extension Definitions, Value Sets and Example Instances are provided. The medical data collected and processed in the RETENTION project are represented in FHIR resources conforming to the specifications described in the Implementation Guide (<https://www.retention-project.eu/ig/index.html>) and exchanged according to the FHIR RESTful API.

This guideline outlines how users can access the repository, perform key operations such as retrieving and uploading models, and troubleshoot common issues. It also details the repository's structure, how the data is structured and formatted, and any key details users should know when interacting with the FHIR repository.

### A.3. Data and Model Sharing

The Data and Model sharing tools are under development. They will be reported in detail in the final version of the system.



#### A.4. Model Specification Tool, Big Data Analytics Engine, Disease Insights

For the Model Specification Tool, instructions are provided for the three identified roles: Data scientist, DevOps, and Clinicians.

The guide for data scientists includes details on how to create, test, and refine Machine Learning artefacts, ensuring they align with the RETENTION platform's standards and requirements.

The DevOps guide covers how to facilitate the deployment and integration of these artefacts within the platform, and how to address any compatibility or performance issues that arise.

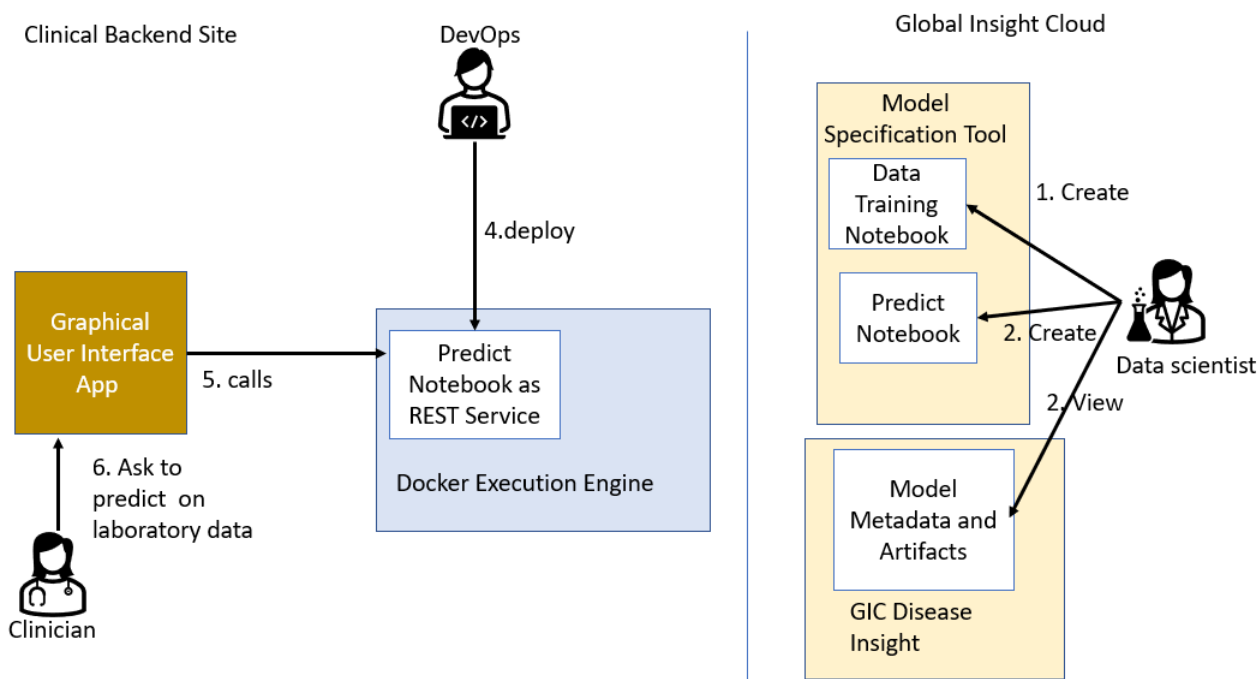
The Clinician guide explains how to interpret the results of the Machine Learning analyses, apply them in a clinical context, and troubleshoot any issues related to viewing or understanding these results.

Notably, as we already provided some instructions for ML models run for third party users in Annex A of D5.1 “RETENTION Analytics & Decision-making enabling mechanisms v1”, we will highlight any critical updates or changes to these guidelines in the present manual. The below user guidelines are still preliminary, they represent the procedure followed during this early phase of the solution development and changes are expected once with the development of RETENTION platform v2 and elaboration of the ML tool.

Within the Machine Learning part of the project, the three roles identified as interacts with the ML tool, are the following:

1. Data scientist who is the main provider of the Machine Learning artefacts
2. DevOps who is the facilitator of the Machine Learning artefacts to the consumers' application
3. Clinician who is the receiver of the results of the Machine Learning

Figure 5 depicts the overview of the interactions of these roles and their activities within the used components:

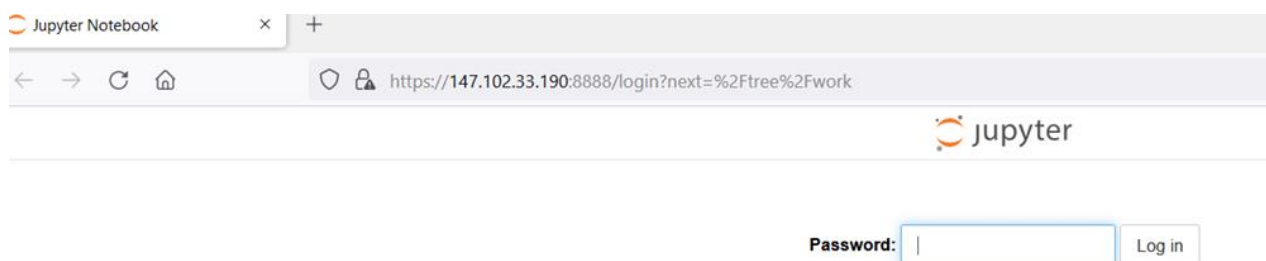


**Figure 5. Roles and their interaction within ML components**

Below we will explain how each of this role can accomplish their purposed tasks.

On the GIC side, the Data Scientist will do the following:

1. Log in into the Model Specification Tool



**Figure 6. Log in for the Model Specification Tool**

2. Create the training notebook, by choosing the preferred programming language, for performing training o dataset and corresponding code for tracking model parameters, KPIs, binary of the model.



Figure 7. Create a new Notebook

3. In cell, add the ML code that performs training o dataset and corresponding code for tracking model parameters, KPIs, binary of the model.

```
# Build the machine Learning model based on the initial data from the article and LogisRegression Algorithm
def hftrain(save_model: False, display_cfm: False):
    def eval_metrics(actual, pred):
        rmse = np.sqrt(mean_squared_error(actual, pred))
        mae = mean_absolute_error(actual, pred)
        r2 = r2_score(actual, pred)
        return rmse, mae, r2

    warnings.filterwarnings("ignore")
    np.random.seed(40)

    # Read the /home/jovyan/work/data/heart-failure.csv file initial dataset from the article
    path_to_data = '/home/jovyan/work/data/heart-failure.csv'
    data = pd.read_csv(path_to_data)

    # Split the data into training and test sets. (0.75, 0.25) split.
    train, test = train_test_split(data)

    train_x = train.drop(["Event", "TIME"], axis=1)
    test_x = test.drop(["Event", "TIME"], axis=1)
    train_y = train[["Event"]]
    test_y = test[["Event"]]

    run_id=None
    model_name="heart-failure"

    # Useful for multiple runs (only doing one run in this sample notebook)
    with mlflow.start_run():
        #use LogisticRegression ML algorithm
        lr = LogisticRegression(random_state = 42)

        #train the model
        lr.fit(train_x, train_y)
        run_id = mlflow.active_run().info.run_id

        # Evaluate Metrics
        predicted_qualities = lr.predict(test_x)
        # Get the probability for the death event only
        predicted_proba=lr.predict_proba(test_x)[:,-1]

        #Get the model performance
        auc_score = roc_auc_score(test_y, predicted_proba)
        (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)

        #Get the confusion matrix of the model
        cf_matrix = confusion_matrix(test_y, predicted_qualities)
        disp = ConfusionMatrixDisplay(confusion_matrix=cf_matrix, display_labels=lr.classes_)
        fig,ax = plt.subplots()
        if (display_cfm == True):
            disp.plot(ax=ax)
            plt.show()

        #Logging the model performance in MLFlow
        mlflow.log_metric('auc', auc_score)
        mlflow.log_metric('rmse', rmse)
        mlflow.log_metric('mae', mae)
        mlflow.log_metric('r2', r2)

        # Logging the confusion matrix of the model
        path=model_name+"-confusion-matrix.png"
        mlflow.log_figure(fig,path)
        print(mlflow.get_artifact_uri(path))
        # Logging the feature importance for the prediction part
```

Figure 8. Sample ML Python code

4. Runs the notebook either by running each cell or by restarting it and run all cells

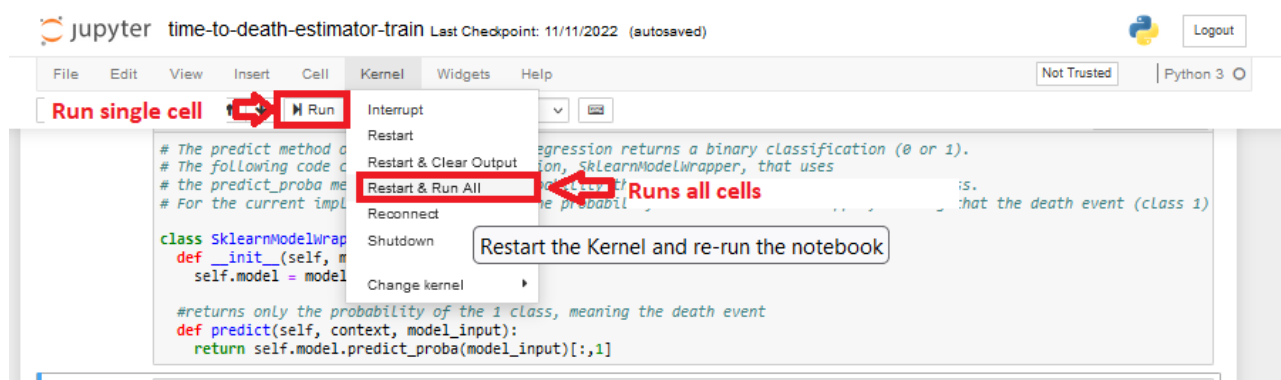


Figure 9. Running the notebook

5. Log in into GIC Disease Insights

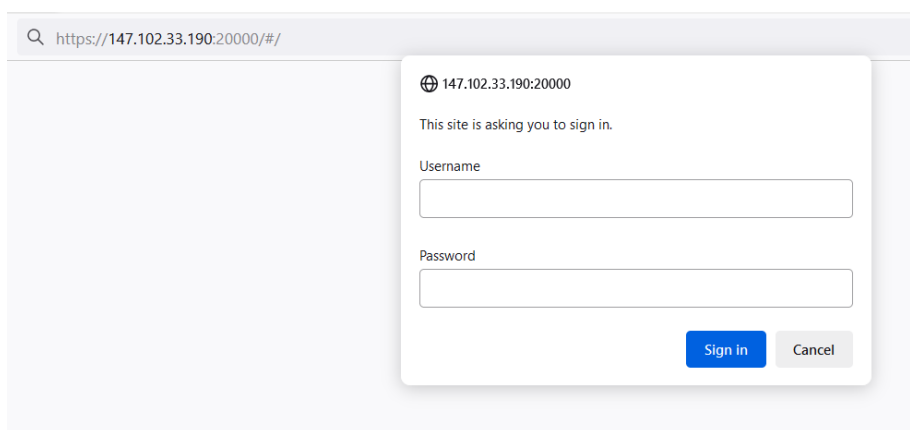


Figure 10. Log in screen for GIC Disease Insights



## 6. View the results of the training into the GIC Disease Insights

The screenshot shows the mlflow Experiments interface. The top part displays a list of runs with columns for Start Time, Duration, Run Name, User, Source, Version, Models, auc, and mae. An arrow points to the 'Start Time' column header with the text 'Click to visualize model details'. Below the list, a detailed view of a specific run is shown, including its date, duration, source, user, status, and lifecycle stage. The 'Metrics (4)' section is expanded, showing a table with columns for Name and Value.

Name	Value
auc	0.914
mae	0.143
r2	0.402
rmse	0.379

Figure 11. List of the performed notebook training experiments

▼ Artifacts

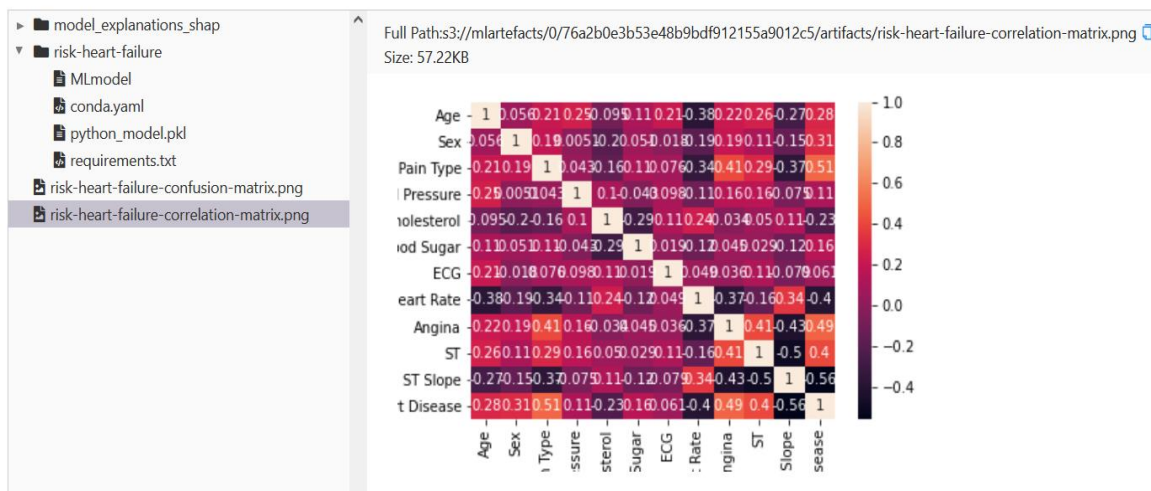


Figure 12. Visualise the model's performance

7. If the KPIs of the model are satisfactory, then he/she will create the prediction notebook that uses the trained model by following the same steps 2,3 described above and adapting the code for prediction

```
in [10]:  
# POST /heartf/predict  
import mlflow.pyfunc  
#from numpyencoder import NumpyEncoder  
model_name = "heart-failure"  
import json  
  
try: REQUEST  
except NameError: REQUEST = None  
  
model_version=1  
patId=100383  
timeAxis=[]  
fData=[]  
dataLoadInst=eval("ICCSFHIRPredictDL1")  
if REQUEST is None:  
  
    fData=dataLoadInst.transform(dataLoadInst.load(patId))  
    data=fData.drop(["Meas_day", "PatId"],axis=1)  
    timeAxis=list(fData['Meas_day'])  
else:  
    req = json.loads(REQUEST)  
    model_version=req['body']['model_version']  
    patId=req['body']['patId']  
    #fData=getPatientData(patId)  
    fData=dataLoadInst.transform(dataLoadInst.load(patId))  
  
    data=fData.drop(["Meas_day", "PatId"],axis=1)  
    timeAxis=list(fData['Meas_day'])  
  
model = mlflow.pyfunc.load_model(  
    model_uri=f"models://{model_name}/{model_version}"  
)  
  
k=("prediction", "day", "xAxisLabel", "yAxisLabel")  
v=(model.predict(data).tolist(),timeAxis, "Measure day", "Death probability")  
predStructure=json.dumps(dict(map(lambda myk,myv: (myk,myv) , k,v)))  
print(predStructure.replace("}",",")+","\n\data\":"+fData.to_json(orient = 'records')+}")
```

Figure 13. Code sample for model prediction

8. Depending on the HTTP request REST access mode, the notebook will be annotated with corresponding Jupyter Kernel Gateway annotation

Sample annotation:

```
# POST /heartf/predict
```

where "POST" is the HTTP REST verb and "/heartf/predict" is the path where the API will be accessible.

This means that the REST API will be accessible for example, [http://csb\\_host\\_machine:host\\_port/heartf/predict](http://csb_host_machine:host_port/heartf/predict) via POST method.





After a prediction notebook was finished by the Data Scientist, the DevOps will perform the following on the CSB side:

9. Securely copy the prediction notebook from the GIC side to the CSB side into the folder `<csb_installation_folder>/retention/jupyter/notebook/retention-api`

Sample copy command:

```
scp siemens@gic_machine:/home/siemens/retention/jupyter/notebook/retention-api/time-to-death-estimator-predict.ipynb .
```

10. If there is a new prediction notebook, then inside the `docker-compose.yml` file he/she will add a new service similar with the content that will be provided further in this section.
11. After editing the `docker-compose.yml` file, DevOps will restart the stack via the command (from the folder `<csb_installation_folder>/retention/screen-mockup`):  
`docker-compose restart`

In order to expose a notebook as REST service, the content that should be added in the `docker-compose.yml` should be similar with:

*services:*

```
siem-an-surv-rate-pred:
  image: spicoflorin/retention-jupyter:rc3
  container_name: "siem-gbc-surv-rate"
  command: >
    jupyter kernelgateway --KernelGatewayApp.allow_headers="*" --
    KernelGatewayApp.api=notebook-http --KernelGatewayApp.ip=0.0.0.0 --
    KernelGatewayApp.allow_origin="*" --
    KernelGatewayApp.seed_uri="/home/jovyan/work/retention-api/time-to-death-estimator-
    predict.ipynb" --debug
  ports:
    - "29000:8888"
  volumes:
    - ../jupyter/notebook:/home/jovyan/work
  restart: always
  env_file: ./config-env/.env.iccs-vm
```

The parts that should be changed are:

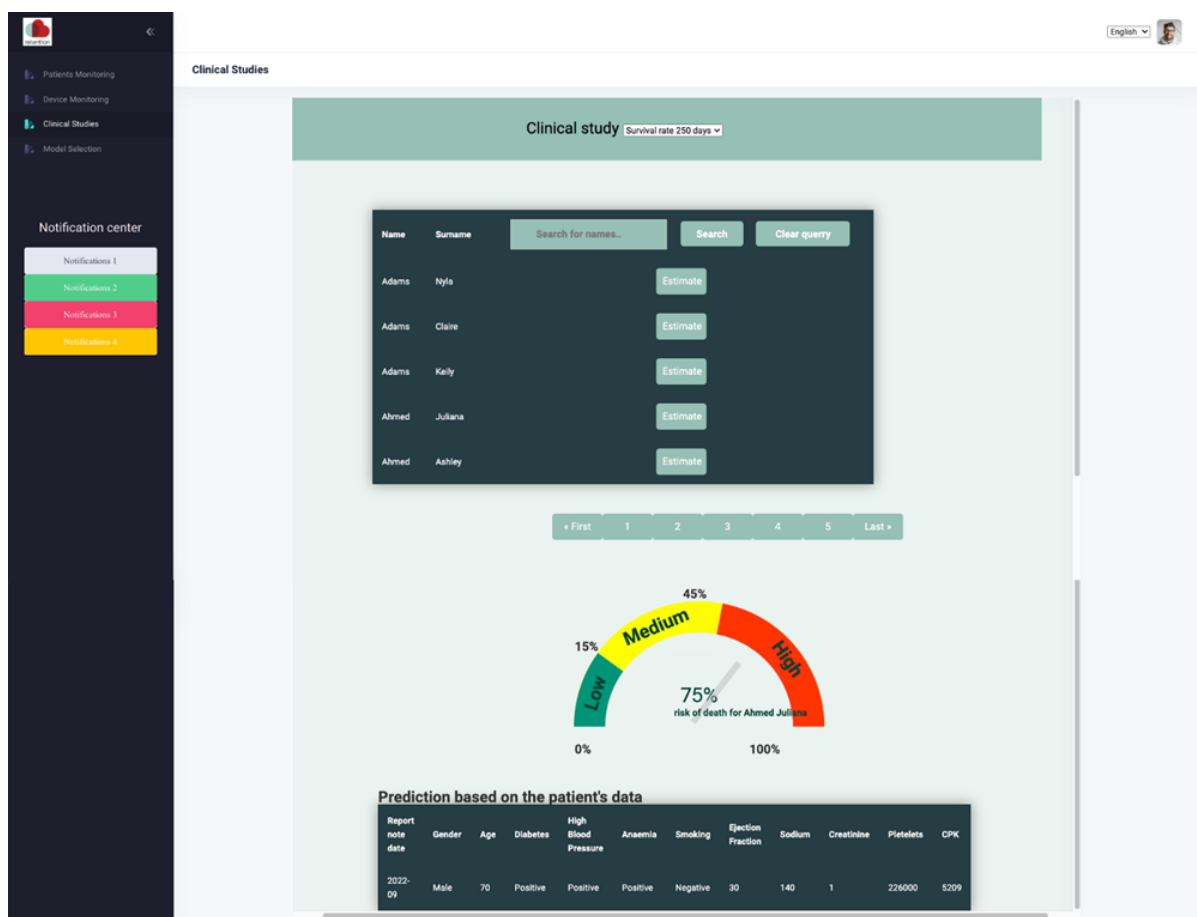
- **container\_name** -> name of the container that should be unique
- **KernelGatewayApp.seed\_uri** -> path of the predict notebook
- **Exposed container port to the outside world <new\_open\_port>:8888**

When the prediction service was ready to be consumed by the Graphical User Interface, on the CSB side, the Clinician will access the outcomes by following the steps:

- Login into the Graphical User Interface

- Go to the Clinical studies
- Select one of the existing clinical studies
- Press the Execute button on one of the patients

The below picture summarizes the above process:



**Figure 14. Sample clinician screen for ML activity**

Validation testing is the process of ensuring that the tested and developed modules meet the user (clinicians or data scientist) needs. Business requirement logic or scenarios must be tested in detail to assure that the critical functionality is covered. To achieve that, a series of methods will be explored and applied within the testing phase.



## A.5. Mobile App

The RETENTION application manual can be found on Basecamp and as a supplementary material (S3\_Retention\_Application\_Manual v0.6):

<https://3.basecamp.com/4281217/buckets/22373922/uploads/6485054451>

Wizards: The RETENTION application is, in its entirety, based on wizards' logic, in order to facilitate ease of use. It essentially guides the user through the necessary actions to be taken by breaking them down into smaller, more manageable parts, each one requiring the user to take some actions or input some data. It presents the end-user with a series of steps to follow, each of them clearly explained.

Initially the app is configured by the Clinical Tech Support. The technical user sets up the database URL so that the device data will be directed to the specific pilot where the patient is enrolled.

For example:

Welcome Screen: Explains the Retention project, what the app does and provides information about the heart failure condition.

Privacy policy: Explains how Retention handles the information and data gathered by the app and asks the user to read and agree to these terms.

Device pairing: Instructions are presented on how to pair with each device.

Measurements: Guides the user and explains how to take measurements.

Data synchronization: After measurements are taken, the app displays the recorded values before they are saved and transmitted to the database.

Questionnaire: A weekly questionnaire that asks the user a few simple questions regarding changes to their condition.

Video tutorials: To be developed

## A.6. Local Home Gateway

Guideline instructions of the Home Gateway are available for both Tech support users performing installations and patients/carers for their set-up and use at home. The Local Home Gateway Installation instructions on how to set up and install RASP-PI are included in the Supplementary Material "Local\_Home\_Gateway\_Installation\_V0.3.2", attached along with this document. Further, the patients/carers manual providing information for patients on how to connect the RASP-PI and sensors at home is provided in the Supplementary Material "Manual for end-users". The documentation containing the installation steps is found at this link (S4\_Local\_Home\_Gateway\_Installation\_V0.3.2):

[https://storage.3.basecamp.com/4281217/blobs/be351970-4511-11ee-8d06-b2d1a80aa527/download/Local\\_Home\\_Gateway\\_Installation\\_V0.3.2.pdf](https://storage.3.basecamp.com/4281217/blobs/be351970-4511-11ee-8d06-b2d1a80aa527/download/Local_Home_Gateway_Installation_V0.3.2.pdf)

The user-manual can be found here (S5\_Home Gateway Manual for users):

<https://storage.3.basecamp.com/4281217/blobs/c8c4c28c-4511-11ee-930c-b2d1a80aa527/download/Manual%20for%20users.docx.pdf>



## Appendix B. Setup instructions

### B.1. MST, BDA, DI: Setup installation steps

For the current DevOps operations own code repository and public docker image repository for the tested docker images were used.

Because the code and docker image repositories, are not yet publicly accessible the installation steps are:

1. manually copy (via *scp* command) the full project code to the GIC and CSB machines
2. push two customized docker images (for Jupyter and MLflow) to public GitHub repository

The current approach of installing the components can change during the next phase of the project, according with the development.

#### B.1.1 GIC layer

On the GIC machine side, the project structure looks as represented in Figure 15:

```
siemens@retention-gic-test:~/retention$ ls -altr
total 80
drwxrwxr-x  2 siemens siemens 4096 Aug  2  2022 postgres-docker
-rw-rw-r--  1 siemens siemens  65 Aug  2  2022 .env
-rwxrwxr-x  1 siemens siemens 1413 Aug  2  2022 iccs-start-stack.sh
drwxrwxr-x  2 siemens siemens 4096 Aug  2  2022 ml-flow-docker
drwxrwxr-x  3 siemens siemens 4096 Aug  2  2022 screen-mockup
drwxrwxr-x  4 siemens siemens 4096 Aug  2  2022 s3-minio-data
-rw-rw-r--  1 siemens siemens 2970 Aug  2  2022 README.md
drwxrwxr-x  2 siemens siemens 4096 Aug  2  2022 minio-s3
drwxrwxr-x  2 siemens siemens 4096 Aug  2  2022 postman-http-request
-rwxrwxr-x  1 siemens siemens  78 Aug  2  2022 createStartStack.sh
-rw-rw-r--  1 siemens siemens  24 Aug  2  2022 .gitignore
-rwxrwxr-x  1 siemens siemens 5167 Aug  2  2022 docker-compose.yaml
drwxrwxr-x  2 siemens siemens 4096 Aug  2  2022 nginx-rev-proxy
drwxrwxrwx  4 siemens siemens 4096 Aug  9 12:45 jupyter
drwxrwxr-x  2 siemens siemens 4096 Oct 18 06:39 config-env
-rw-rw-r--  1 siemens siemens  335 Oct 18 06:48 .env.iccs-vm
-rw-rw-r--  1 siemens siemens 3107 Dec  9 09:43 iccs-docker-compose.yaml
drwxr-x--- 10 siemens siemens 4096 Feb  3 08:40 ..
drwxrwxr-x 11 siemens siemens 4096 Feb  3 08:44 .
```

Figure 15. The project structure of the model specification tool

To start the whole analytical platform, from the `/home/siemens/retention` folder, the following command must be used,

```
sh iccs-start-stack.sh
```

To verify that the platform is up and running, from the `/home/siemens/retention` folder, the following command must be executed:

```
docker ps
```

A successful start is depicted by Figure 16.

```

siemens@retention-gic-test:~/retention$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
fc3e9bc6d8b6   spicoflorin/retention-jupyter:rc3   "tini -g -- start-no..." 7 weeks ago   Up 7 weeks    0.0.0.0:8888->8888/tcp    jupyter-sie
773744278ae2   spicoflorin/retention-mlflow:rc2    "./wait-for-it.sh po..." 7 weeks ago   Up 7 weeks    5000/tcp                 app-server-mlflow-track
ing
4115c1c92bb1   minio/mc:RELEASE.2021-12-20T23-43-34Z "/bin/sh -c ' while ..." 7 weeks ago   Up 7 weeks                                retention_createbuckets
1
5b08c48b5bda   postgres:14                          "docker-entrypoint.s..." 7 weeks ago   Up 7 weeks    5432/tcp                 db-mlflow-tracking
fb408dd577d5   minio-s3:rc2                          "/usr/bin/docker-ent..." 7 weeks ago   Up 7 weeks    0.0.0.0:9500->9000/tcp    retention_s3-minio-simu
lator_1
fb36c74bc52d   reverseproxy:rc1                      "/bin/bash /app/entr..." 7 weeks ago   Up 7 weeks    0.0.0.0:20000->12443/tcp  nginx-rev-proxy

```

Figure 16. The successful start of the docker image

Figure 17 summarizes the started containers and their dependencies:

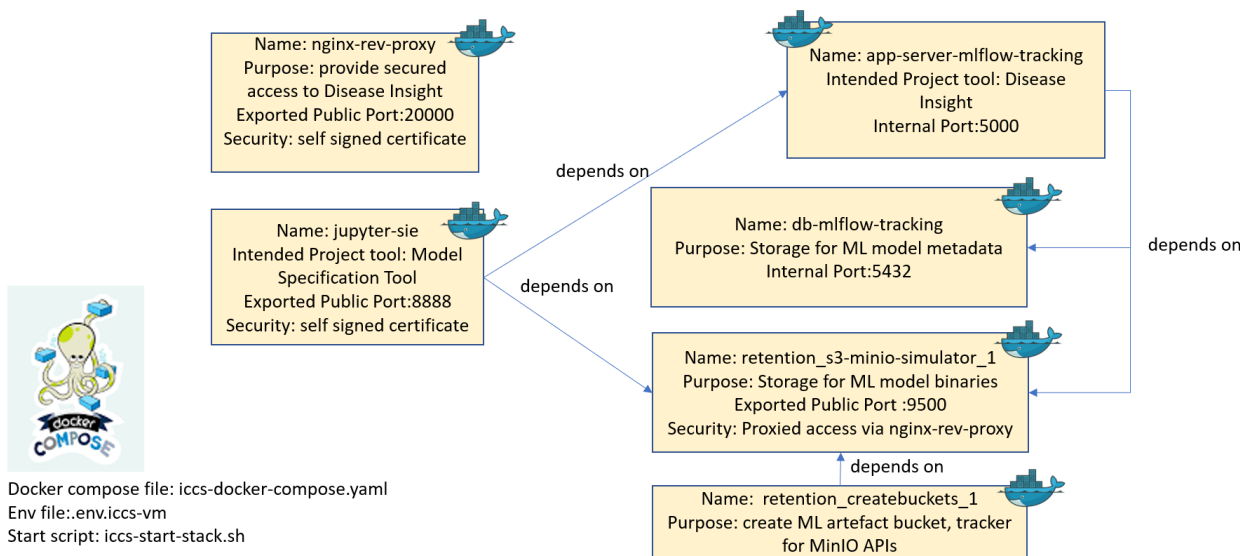


Figure 17. Details of the started containers in the analytical stack on GIC

The *iccs-docker-compose.yaml* file depends on the *env.iccs-vm* file. The content of this file as long with the explanations can be found in the Appendix C.1.1.

### B.1.2 CSB layer

On the CSB machine side, a simpler version of the GIC version is used, that contains only two folders in the */home/siemens/retention* directory:

```

siemens@retention-gsb-test:~/retention$ ls -ltr
total 16
drwxrwxr-x 3 siemens siemens 4096 Jul 20 2022 jupyter
drwxrwxr-x 4 siemens siemens 4096 Oct 22 16:36 .
drwxrwxr-x 3 siemens siemens 4096 Dec 12 10:05 screen-mockup
drwxr-x--- 7 siemens siemens 4096 Feb 3 09:03 ..
siemens@retention-gsb-test:~/retention$

```

Figure 18. Project structure for the CSB side

In order to start the prediction service and the consuming web application, on the folder */home/siemens/retention/screen-mockup*, we execute the command:

**docker-compose up**

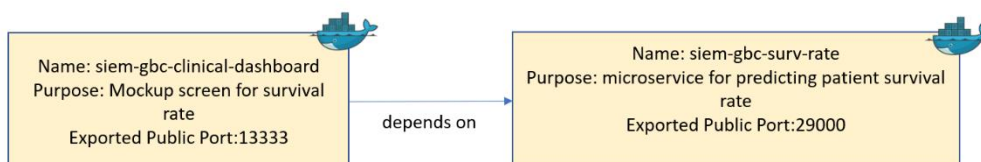
The started docker containers of the above command, can be seen by executing the following command:

**docker ps | grep siem-**

```
siemens@retention-gsb-test:~/retention/screen-mockups$ docker ps | grep siem
fc65e89c5aaa  spicoflorin/retention-jupyter:rc3          "tini -g -- jupyter ..." 7 weeks ago  Up 7 weeks      0.0.0.0:290
00->8888/tcp          siem-gbc-surv-rate
2cc2ecbf43be  nginx:1.23-alpine                          "/docker-entrypoint..." 7 weeks ago  Up 7 weeks      0.0.0.0:133
33->80/tcp           siem-gbc-clinical-dashboard
```

Figure 19. Validation of images running on CSB

Detailed view of the started containers is depicted by the following image:



Docker compose file: docker-compose.yaml  
Env file: config-env/.env.iccs-vm  
Start script: docker-compose up

Figure 20. Details of the started containers in the analytical stack on CSB

This stack is referring to the config-env/.env.iccs-vm environment file that can be found in the Appendix C.1.2. The contained environment variables should be reviewed for each CSB installation, mainly the variable FHIR\_SURV\_RATE\_API\_URL that should point out to the local FHIR server.

The associated yaml and environment files for the afore mentioned environments (GIC and CSB) can be found in the Appendix C.1 BDA Docker-compose files.

### B.1.3 Accessing the REST API

To access the MLflow models for predicting patient's evolution in future for one scenario this is an example of using the REST API exposed through MLFLOW:

```
curl --location 'http://147.102.33.191:29000/heartf/predict' \
--header 'Content-Type: application/json' \
--data '{
  "model_version":1,
  "patId":100383
}'
```

A potential result is the following:

```
{"prediction": [0.14228677154675975, 0.051550792140588324, 0.3770164969045237,
0.2800791373795637, 0.19060923818073525], "day": ["2022-01", "2022-03", "2022-06", "2022-08", "2022-
09"], "xAxisLabel": "Measure day", "yAxisLabel": "Death
probability", "data": [{"PatId": "Patient/100383", "Gender": 0, "Smoking": 0, "Diabetes": 0, "BP": 0, "Anaemia": 0,
"Age": 49, "Ejection.Fraction": 38.0, "Sodium": 141.0, "Creatinine": 1.2, "Pletelets": 147000.0, "CPK": 582.0, "Meas_
```



```
day": "2022-01"}, {"PatId": "Patient\\100383", "Gender": 0, "Smoking": 0, "Diabetes": 0, "BP": 0, "Anaemia": 0, "Age": 49, "Ejection.Fraction": 60.0, "Sodium": 137.0, "Creatinine": 1.0, "Platelets": 242000.0, "CPK": 1610.0, "Meas_day": "2022-03"}, {"PatId": "Patient\\100383", "Gender": 0, "Smoking": 0, "Diabetes": 0, "BP": 0, "Anaemia": 0, "Age": 49, "Ejection.Fraction": 35.0, "Sodium": 132.0, "Creatinine": 1.0, "Platelets": 243000.0, "CPK": 5882.0, "Meas_day": "2022-06"}, {"PatId": "Patient\\100383", "Gender": 0, "Smoking": 0, "Diabetes": 0, "BP": 0, "Anaemia": 0, "Age": 49, "Ejection.Fraction": 25.0, "Sodium": 136.0, "Creatinine": 1.6, "Platelets": 252000.0, "CPK": 369.0, "Meas_day": "2022-08"}, {"PatId": "Patient\\100383", "Gender": 0, "Smoking": 0, "Diabetes": 0, "BP": 0, "Anaemia": 0, "Age": 49, "Ejection.Fraction": 35.0, "Sodium": 132.0, "Creatinine": 1.0, "Platelets": 222000.0, "CPK": 582.0, "Meas_day": "2022-09"}]}
```

This is one sample of how to use the CSB layer to predict patients' information using models previously created in the GIC layer.

As the project evolves and patient's data will be injected in the platform, we will develop more elaborated models in GIC, exposing in the CSB layer more AI functionalities to be later used by clinicians.



## Appendix C. Docker-compose files

### C.1 MST, BDAE, DI docker-compose files

Docker compose files for each hospital instance backend, as well as the dashboard, can be found here: <https://3.basecamp.com/4281217/buckets/22373922/uploads/6485182832>

Note: For security reasons, sensitive environment variables have been removed.

#### C.1.1. MST, BDAE, DI Docker-compose on GIC side

**Name:** iccs-docker-compose.yaml

**Content:**

```
version: '3'
services:
  notebook:
    image: spicoflorin/retention-jupyter:rc3
    container_name: "jupyter-sie"
    ports:
      - "8888:8888"
    volumes:
      - ./jupyter/notebook:/home/jovyan/work
      - ./jupyter/config:/home/jovyan/.jupyter

    depends_on:
      - mlflow
      - s3-minio-simulator
    environment:
      GEN_CERT: 'yes'
      NB_UID: 1000
    env_file:
      - .env.iccs-vm
    restart: always

  reverseproxy:
    image: reverseproxy:rc1
    build:
      context: ./nginx-rev-proxy
```





```
container_name: "nginx-rev-proxy"
```

```
ports:
```

```
  - 20000:12443
```

```
restart: always
```

```
env_file:
```

```
  - .env.iccs-vm
```

```
mlflow:
```

```
image: spicoflorin/retention-mlflow:rc2
```

```
hostname: mlflow
```

```
restart: always
```

```
container_name: "app-server-mlflow-tracking"
```

```
environment:
```

```
  MLFLOW_S3_ENDPOINT_URL: http://s3-minio-simulator:9000
```

```
  MLFLOW_TRACKING_INSECURE_TLS: 'true'
```

```
expose:
```

```
  - "5000"
```

```
depends_on:
```

```
  - postgres
```

```
  - s3-minio-simulator
```

```
command: >
```

```
  ./wait-for-it.sh postgres:5432 -- mlflow server
```

```
  --backend-store-uri postgresql://mlflow:secret@postgres:5432/mlflow
```

```
  --default-artifact-root s3://mlartefacts/
```

```
  --host 0.0.0.0
```

```
postgres:
```

```
image: postgres:14
```

```
container_name: db-mlflow-tracking
```

```
restart: always
```

```
user: postgres
```

```
volumes:
```

```
  - store-pg:/var/lib/postgresql/data
```

```
  - ./postgres-docker/mlflow_schema_with_one_registered_model.sql:/docker-entrypoint-initdb.d/mlflow_schema_with_one_registered_model.sql
```



s3-minio-simulator:

build:

context: ./minio-s3

restart: always

image: minio-s3:rc2

ports:

- "9500:9000"

volumes:

- ./s3-minio-data:/data

command: server /data

user: \${CURRENT\_UID}

createbuckets:

image: minio/mc:RELEASE.2021-12-20T23-43-34Z

depends\_on:

- s3-minio-simulator

entrypoint: >

/bin/sh -c "

while ! curl -s s3-minio-simulator:9000 > /dev/null; do echo waiting for minio; sleep 30; done;

/usr/bin/mc alias set myminio http://s3-minio-simulator:9000;

/usr/bin/mc mb --ignore-existing myminio/mlartefacts;

mc admin trace myminio -v;

"

restart: on-failure

volumes:

store-pg:

Name: .env.iccs-vm

Content:

#URL for the Disease Insight

MLFLOW\_TRACKING\_URI=https://reverseproxy:12443

#URL for the Disease Insight binaries storage

MLFLOW\_S3\_ENDPOINT\_URL=http://s3-minio-simulator:9000



```
#Disease Insight user (Model tracking UI and API)
MLFLOW_TRACKING_USERNAME=
#Disease Insight (Model tracking UI and API) password
MLFLOW_TRACKING_PASSWORD=
#Allow self-signed certificate access to Disease Insight
MLFLOW_TRACKING_INSECURE_TLS=true
#Allow insecure access to Disease Insight ML storage for binaries (MinIO)
MLFLOW_S3_IGNORE_TLS=true
#Disease Insight binaries storage credentials (MinIO)
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
```



### C.1.2. BDAEE Docker-compose on CSB side

**Name:** docker-compose.yaml

**Content:**

```
version: '3'

services:

  siem-an-surv-rate-pred:

    image: spicoflorin/retention-jupyter:rc3

    container_name: "siem-gbc-surv-rate"

    command: >

      jupyter kernelgateway --KernelGatewayApp.allow_headers="'*'" --
      KernelGatewayApp.api=notebook-http --KernelGatewayApp.ip=0.0.0.0 --
      KernelGatewayApp.allow_origin="'*'" --
      KernelGatewayApp.seed_uri="/home/jovyan/work/retention-api/time-to-death-estimator-
      predict.ipynb" --debug

    ports:

      - "29000:8888"

    volumes:

      - ../jupyter/notebook:/home/jovyan/work

    restart: always

    env_file: ./config-env/.env.iccs-vm

  siem-an-clinical-dashboard:

    image: nginx:1.23-alpine

    container_name: "siem-gbc-clinical-dashboard"

    ports:

      - 13333:80

    restart: always

    volumes:

      - ./index.html:/usr/share/nginx/html/index.html
```

This stack is referring to the config-env/.env.iccs-vm environment file.

**Name:** config-env/.env.iccs-vm

**Content:**

```
#URL for the GIC Disease Insight
MLFLOW_TRACKING_URI=
#URL for the GIC Disease Insight binaries storage
```



```
#Disease Insight user(GIC Model tracking UI and API)
MLFLOW_TRACKING_USERNAME=
#Disease Insight (GIC Model tracking UI and API) password
MLFLOW_TRACKING_PASSWORD=
#Allow self-signed certificate access to Disease Insight
MLFLOW_TRACKING_INSECURE_TLS=true
#Allow insecure access to GIC Disease Insight ML storage for binaries (MinIO)
MLFLOW_S3_IGNORE_TLS=true
#GIC Disease Insight binaries storage credentials (MinIO)
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=

##FHIR API Local Database to get the patient data for survival rate. Currently, Dec 2022,
##for demo purpose only use the Global Infra Cloud FHIR until the integration will be performed
FHIR_SURV_RATE_API_URL=http://<fhir_csb_file>
/fhir/Patient?_id={id}&_revinclude=Observation:subject&_revinclude=Condition:subject
```