Project no. 826278

SERUMS

Research & Innovation Action (RIA)
SECURING MEDICAL DATA IN SMART-PATIENT HEALTHCARE SYSTEMS

Report on Final Smart Health Centre System Software
D6.3

Due date of deliverable: 30th April 2022

Start date of project: 1st January 2019

Type: Deliverable
WP number: WP6

Responsible Institution: University of St Andrews
Editor and editor's address: Thais Webber (tcwds@st-andrews.ac.uk)

Reviewers: Juliana Bowles

Version 1.0

Project co-founded by the European Commission within the Horizon H2020 Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

1 Release History

Release No.	**Dates**	**Author(s)**	**Release Description/Changes made**
V0.1	15/4/2022	Agastya Silvina (USTAN) Matthew Banton (USTAN) Thais Webber (USTAN) Andrew Bowles (USTAN) Argyris Constantinides (UCY)	Final report on the Serums Smart Health Centre System (SHCS) integration and testing (architectural aspects of SHCS and APIs and WUI design). Added report from PoC3 execution at USTAN.
V0.2	28/04/2022	Eduard Baranov (UCL) Thomas Given-Wilson (UCL) Argyris Constantinides (UCY) Thais Webber (USTAN)	Added sections concerning the SHCS modelling and security properties verification. Added details on the testing performed.
V0.3	29/04/2022	Juliana Bowles (USTAN) Andrew Bowles (USTAN)	Final review

2 Serums Consortium

Partner 1	**University of St Andrews**
Contact Person	Name: Juliana Bowles Email: jkfb@st-andrews.ac.uk
Partner 2	**Zuyderland Medisch Centrum**
Contact Person	Name: Larissa Haen-Jansen Email: la.jansen@zuyderland.nl
Partner 3	**Accenture B.V.**
Contact Person	Name: Bram Elshof Email: bram.elshof@accenture.com
Partner 4	**IBM Israel Science & Technology Ltd.**
Contact Person	Name: Michael Vinov Email: vinov@il.ibm.com
Partner 5	**Sopra-Steria**
Contact Person	Name: Andre Vermeulen Email: andreas.vermeulen@soprasteria.com
Partner 6	**Université Catholique de Louvain**
Contact Person	Name: Axel Legay Email: axel.legay@uclouvian.be
Partner 7	**Software Competence Centre Hagenberg**
Contact Person	Name: Michael Rossbory Email: michael.rossbory@scch.at
Partner 8	**University of Cyprus**
Contact Person	Andreas Pitsillides

	Email: andreas.pitsillides@ucy.ac.cy
Partner 9	**Fundació Clínic per a la Recerca Biomèdica**
Contact Person	Name: Santiago Iriso Email: siriso@clinic.cat
Partner 10	**University of Dundee**
Contact Person	Name: Vladimir Janjic Email: VJanjic001@dundee.ac.uk

Table of Contents

3 Executive Summary

Securing Medical Data in Smart Patient-Centric Healthcare Systems (Serums) is a research project supported by the European Commission (EC) under the Horizon 2020 program. This document is the third and final deliverable of Work Package 6: “Integration and Testing”. The leader of this work package is USTAN, with involvement from all other partners: ZMC, ACC, IBM ISRAEL, SOPRA STERIA, UCL, SCCH, UCY, FCRB and UNIVDUN.

The purpose of this work package is to integrate the Serums technologies into a coherent Smart Health Centre System (SHCS) that will be used as a central access point to the different techniques developed in the course of the project. We have developed a front-end for the SHCS which considers different perspectives from which the data can be accessed, e.g. patient, specialist, and administration (T6.1). Using input from all WPs, we have integrated smart patient health records, authentication system, authorisation mechanisms using blockchain and data lake, data analytics functionality, and a secure and privacy-preserving communication infrastructure (T6.2). Over the course of the integration, we have also tested the interoperability of the Serums technologies on synthetic data that is produced by the data fabrication mechanisms from WP4 (T6.3). The system testing task in this work package was performed from the developers’ perspective, including testing the front-end, i.e., the user interaction interface features (WUI - *Web User Interface*) and the backend integration, e.g., the APIs calls and responses actions on SHCS. In WP7, in contrast, we evaluate the use cases from the user’s perspective, considering the three refined Use Cases (UCs).

4 Introduction

4.1 Role of the Deliverable

This deliverable entitled “*Report on Final Smart Health Centre System Software*” is the third deliverable of WP6. The deliverable D6.3 reflects the development phase and refinement of the design work performed on D6.1 and D6.2, including front-end and backend design for the integration of Serums technologies within the Serums platform, the *Smart Health Centre System* (SHCS).

USTAN leads this task modifying the proof-of-concept (PoC) system developed in D6.1 and refined it in D6.2 to accommodate the necessary interoperation of the tools and techniques. From initial work (T6.1), we have integrated into the SHCS the technologies developed in WP2–WP5, analysing the interoperability among these technologies, and identifying the issues arising from their design and/or required privacy/security regulations such as GDPR (please refer to deliverable D6.2). SOPRA, SCCH, IBM and UCY have contributed throughout with interoperability aspects of their respective tools/algorithms developed in WP2, WP3, WP4 and WP5, while ZMC and FCRB have contributed with insights arising from realistic scenarios of their use cases and identifying missing requirements concerning user feedback. We have refined the architectural design, describing its development phase and reporting the steps required for the system integration and testing.

Finally, in D6.3 we present the last version of the Serums system, and report all activities related to integration, testing and formal verification of system properties. We focused on the integration aspects bringing the end-user features, the deployment infrastructure, detail on component interconnections/interactions, as well as validation and verification steps for the integrated system.

4.2 Relationship to Other Serums Deliverables

WP6 entitled "Integration and Testing" brings together work done across all WPs of the project. Overall, WP6 consequently integrated smart patient records and authorisation schemes (WP2), data analytics mechanisms (WP3), secure and privacy-preserving communication infrastructure (WP4), and authentication mechanisms (WP5) into the Serums Smart Health Centre System (SHCS). Over the course of the integration, we tested the interoperability of the Serums technologies on synthetic data that is produced by the data fabrication tool (WP4). Use cases (WP7) provide user and system requirements as input, so WP6 improves the coding and integration of functionalities considering overall system design. **Figure 1** shows the overview of the Serums work packages and their dependencies for the project execution. The deliverable D6.3 describes in detail the development phase of the SHCS and it is associated with **Milestone MS14**: *Final Smart Health Centre System, use cases and evaluation, future roadmap and end of the project*, contributing to the Final Smart Health Centre System (SHCS). Further deliverables in WP7 will contribute to the remaining aspects and are due to be reached by M42.

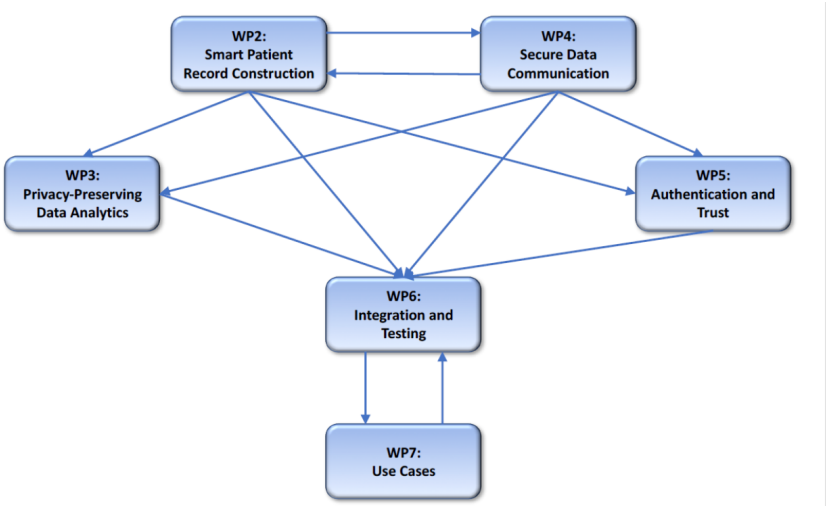


Figure 1 Overview of the Serums work packages and dependencies (PERT chart).

4.3 Structure of this Document

This document D6.3 describes the SHCS design and integration, focusing on the refined system architecture (**Section 5**), the last included features on the front-end, and overall development and testing tasks for the backend performed for the T6.2 and T6.3 providing a system version (**Section 6**). We mention other WPs' contributions for the integration task throughout this document to highlight the joint effort for the architectural design and SHCS/APIs development. The produced software artifacts during the development phase also incorporate the responsibilities of each partner (WP) as well as interfaces and steps required for a seamless SHCS integration and testing.

This deliverable D6.3 also includes details about the system validation step through system testing approach, use cases refinement and respective testing scenarios, and the different descriptive security assessments and evaluations that were published in scientific conferences since the beginning of the project (**Section 6**). In addition, considering the verification step, we report the formal model proposed for the SHCS, which employs timed automata description models and the choice of the UPPAAL model checker tool (**Section 7**). The formal model allowed us to perform formal verification

of security properties about the system, describing all SHCS components as sub-models and a sub-model representing user behavior based on the specifications of the developed system.

Finally, we present in D6.3 document the conclusion (**Section 8**) and next steps for introducing Serums technology in future production environments. We included in **Appendix I** the essential design artifact (Information Flow viewpoint) of the architecture; **Appendix II** contains the Serums Web User Interface (WUI) detail for all system features and functionalities; **Appendix III** presents information on the software libraries included for the integration of the system.

5 Serums Integrated System: architectural design

The Serums tool-chain description, firstly published in a paper [1] (in 2019), describes the Serums project proposal considering the role of each technology and research challenges posed when integrating them in a multinational data sharing platform. The proposed system [2] aims to securely bring healthcare data from different healthcare providers, in a user-friendly way, to authorised individuals and organisations logged in the system.

The Serums platform, specifically, is a secure multi-layered and decentralised patient-centric system built for secure transnational data exchanges [3]. It is composed of a front-end and a backend component. The front-end enables end-users to access the developed technologies in the backend as to be tested and validated by them. Furthermore, the front-end (integrating the backend) plays an important role in validating important requirements of the system, for instance:

- First, patients to have more control over their personal medical data since they can access own medical records and they can also define access privileges to other parties, considering which categories of their medical data they would like to share and for how long [4];
- Second, healthcare professionals can leverage better-informed decisions based on a more holistic view of patients' health records, *i.e.*, professionals can search for a patient and securely retrieve their medical records due to the authorisation scheme in place [5]. They can also issue requests directed to a patient for an access rule creation; patients can then accept (or reject) these rules requests in the system.

Figure 2 presents the Serums platform proposal including specification of its users, modular components and their connections. The Serums platform design has a special concern regarding the interaction of end-users (patients, healthcare professionals, administrators) with the system, thus we followed a user-centred design approach based on three different *Use Cases* (UCs). Supporting a synergistic design process, the UCs include patients' journeys from the Zuyderland Medisch Centrum (ZMC) in the Netherlands, Hospital Clínic de Barcelona (HCB) in Catalonia, and from the Edinburgh Cancer Data (ECD) in Scotland. They have allowed the engagement of several potential users and researchers to construct functional scenarios for diverse interaction situations, guiding system requirements and architectural design. The dashed box in the figure contains the different components of Serums technologies and their interconnections: *Authentication system* (associated to WP5), the *Blockchain component* (associated to WP2), the *Data Lake component* for the SPHR storage and retrieval (associated to WP2), the generation of synthetic data with IBM's *Data Fabrication Platform* (DFP, associated to WP4), and the *privacy-preserving machine learning* models developed and evaluated over the fabricated datasets (associated to WP3). Externally, the three UCs (WP7) are connected to the Data Lake to illustrate that they provide requirements for the SPHR design as a universal format for patients records, and guide the specification of all system functionalities.

The proof-of-concept (PoC) evaluations with end-users (WP7) have used the Serums front-end (web-based system with end-users shown at the left), which interconnects (through the APIs) all the backend components. Following, we detail these components' roles, interconnections and their core processes.

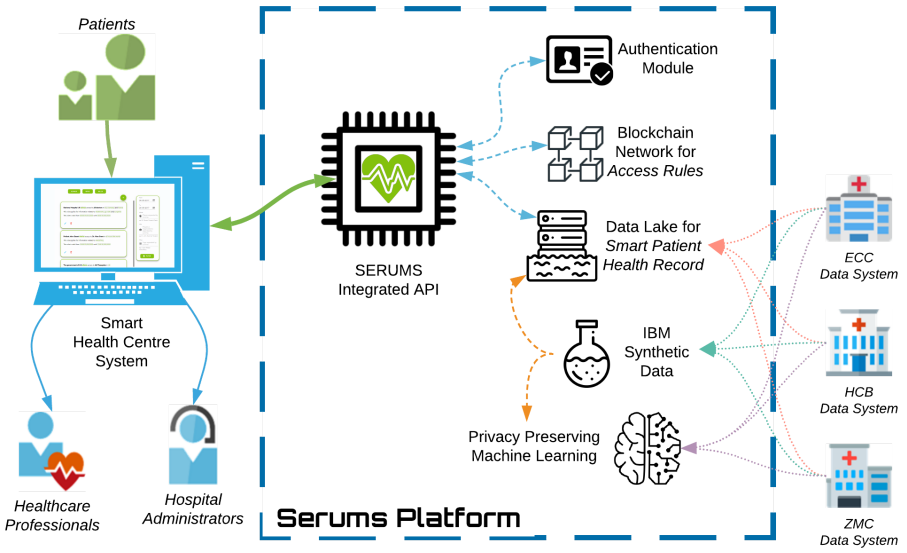


Figure 2 An overview of the Serums tool-chain design.

The *Smart Health Centre System* (SHCS) represents the central point of integration and access of all Serums technologies (The SERUMS Integrated API). The SHCS is built in such a way it can integrate seamlessly and securely the proposed technologies as well as any further customised API to securely access different formats of medical data in future (*e.g.*, from health devices, trackers, other hospital information systems, etc.) [2,3,4,5,6].

The platform integrates and aggregates the generated patient synthetic data (IBM, WP4) [7] in a flexible Data Lake (WP2) as *Smart Patient Health Records* (SPHR). The use of synthetic data allows us to further demonstrate and test the overall tool-chain operation under different scenarios. The concept of SPHR is a universal format for representing the medical data in a graph-based structure and to enable efficient data retrieval proposed by Serums (WP2/WP6). The format integrates healthcare metadata compatible with patients' medical records information present in different distributed sources of medical data (*e.g.*, hospitals and out-of-hospital environments) across Europe (WP2) [6]. While the patient records are centralised as SPHR, the data in them may potentially refer in future to databases distributed inside and outside healthcare environments. These medical records may contain all information about the patients, from static information such as date of birth, gender, and contact information, to vital information such as weight, body mass index, allergies; and dynamic information, for example, about their treatments, appointments, and examinations. For instance, in future some data may be collected from within a healthcare system over trusted networks, while others may be collected from personal health monitoring devices. Data sent over untrusted networks can be secured using Data Encryption mechanisms such as SSL/TLS or RSA (which is already implemented) assuming that the data volume is not too large.

About the Serums components integration depicted in **Figure 2**, when users access patient data, they are first redirected to the Authentication client application (WP5). Once successfully identified, the user is then redirected back to the SHCS client. In the Serums project, our aim is to develop personalised and adaptive multi-factor user authentication schemes [8]. Once the user logs in, their access privileges are checked using the Blockchain component in the backend (WP2).

Different users (e.g., patients, healthcare professionals) have different levels of permissions stored in the blockchain, depending on the access rules created to and by patients, compliant with GDPR[1] and also compliant with other legal and ethical regulations depending on data location. For

[1] Information on GDPR can be found at https://gdpr-info.eu/

example, the patient has access to all available records the system can retrieve, while a healthcare professional can only access parts of the record that are relevant to them. The Blockchain component ensures that only authorised parties can access the medical data, and depending on permissions, possibly only be part of the data. The Blockchain contains the access rules, enables and data access transactions, and keeps a record of data access attempts. Note, however, that no patient data is stored in the Blockchain. Once the user is authenticated and the rules are checked, the requested data from the SPHR in the Data Lake is sent back to the user. The access transaction itself is stored in the Blockchain database.

We published a scientific paper [3] to discuss the Serums architectural design workflow and how we could address the different viewpoints for the system architecture towards the desired quality attributes. In the first SHCS design iteration, we proposed that these different viewpoints are important artifacts to gain an understanding of the required components of the SHCS, as previously discussed in the D6.1 report. The second iteration of the SHCS design and integration has provided refinements on these viewpoints (i.e., the designed artifacts) as well as a scheme of the system's architecture integration and deployment, using the proposed technologies (deliverable D6.2).

Following, for this deliverable (D6.3), we include the latest update on these software artifacts and a discussion on the design decisions throughout the text, highlighting the actions to improve system security, resilience, reliability, and usability aspects concerning the Serums front-end design.

5.1 Context Viewpoint

The Context Viewpoint (**Figure 3**) presents the context for all the components in the Serums platform. It shows how the different users will interact with the *Web User Interface* (WUI) of the SHCS as the main access point and the *Authentication web client* for users authentication.

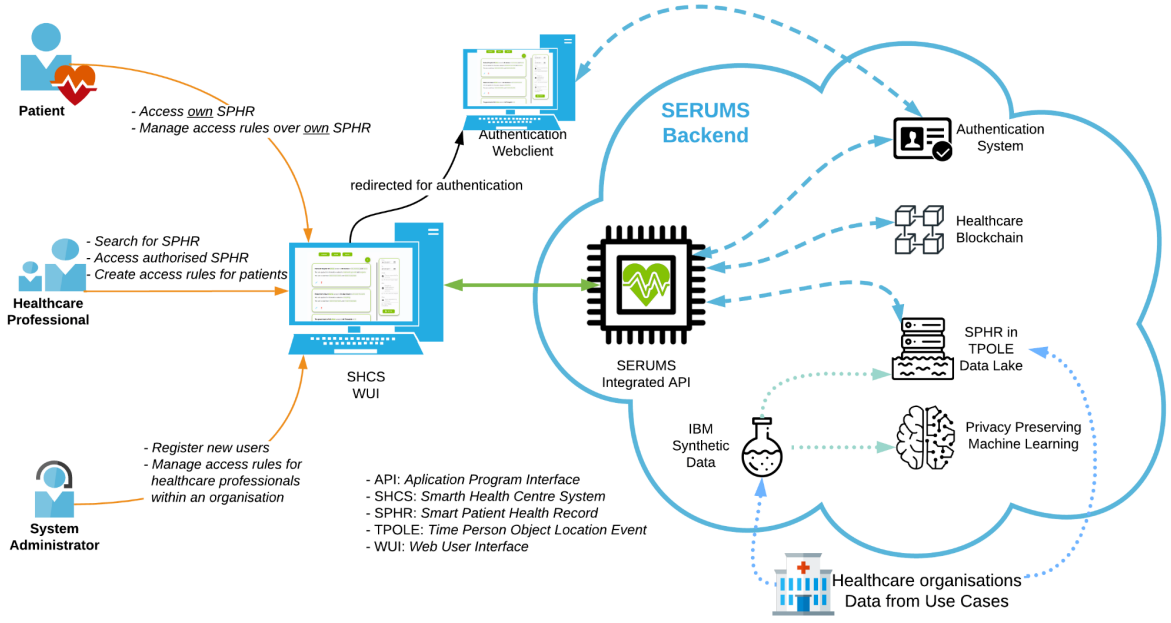


Figure 3 Updated Context Viewpoint

The WUI connects to the integrated components, the Serums API, which forwards all the requests to the internal components, including the *Authentication system* (WP5), the *Blockchain component* (WP2/WP5), and the *Data Lake component* (WP2).

The end-user does not have any direct interaction with the *IBM data synthesizer module* (WP4) or the *Privacy-preserving machine learning module* (WP3), because these functionalities are

reserved for the research and development team only. However, there is a connection between the Data Lake and the generated patient synthetic data, which allows us to further demonstrate and evaluate the overall platform operation, its features, the crucial security aspects.

5.2 Functional Viewpoint

The Functional Viewpoint (**Figure 4**) details the functionalities of each component, and how the integration modules (SHCS and Serums API) connect to the other components via these APIs. This architectural decision was made to enforce decoupling between components, reducing the chances of systemic failures, while easing the integration task. The functionalities are presented according to the requirements elicitation step performed by the WP7 partner for each Use Case (UC), i.e., USTAN, ZMC and FCRB.

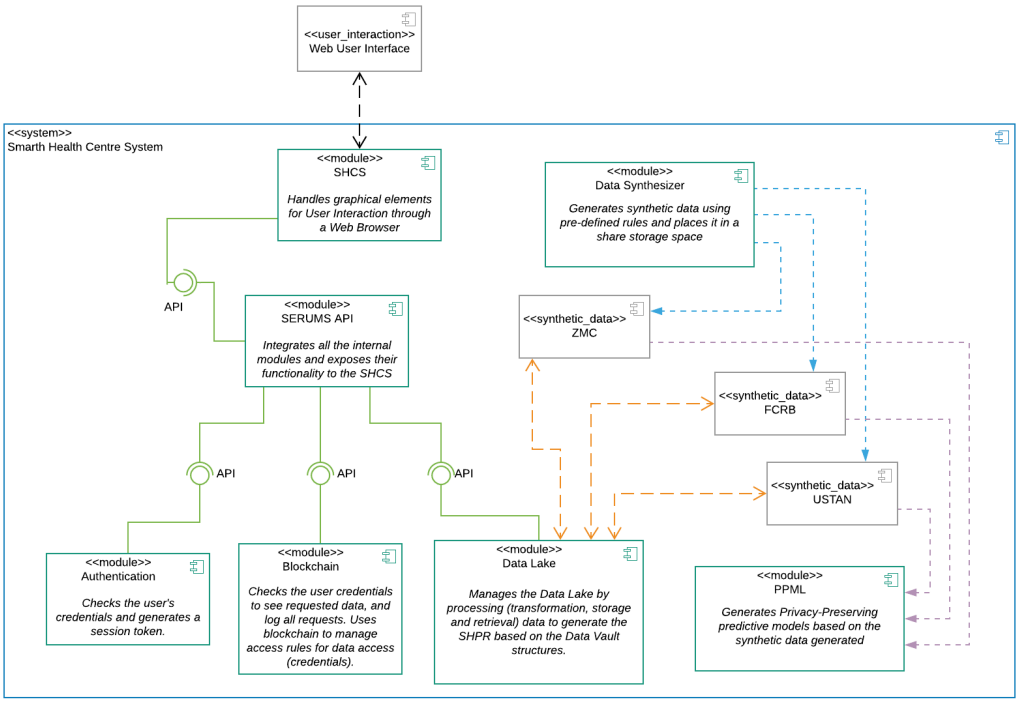


Figure 4 Functional Viewpoint

5.3 Interaction Viewpoint

The Interaction Viewpoint (**Figure 5**) explains the connection between several different main components of the Serums platform. The (User) Authentication system can provide mechanisms to register new users, activate registered users, and to authenticate active users within the system.

The Blockchain component handles the creation, deletion and update of access rules, and the log of events occurred in the system. Finally, the Data Lake component provides a service to retrieve the Smart Patient Health Record (SPHR) at each request, which automatically integrates synthetic data into the proposed SPHR unified format. The current PoC3 system version integrates users (patient) data for each UC.

The Serums API works as a bridge between the SHCS and the other modules as explained above (Figure 5). If the components of the platform are deployed in different locations, it would be the main feature of the Serums API to connect and access these components from different locations.

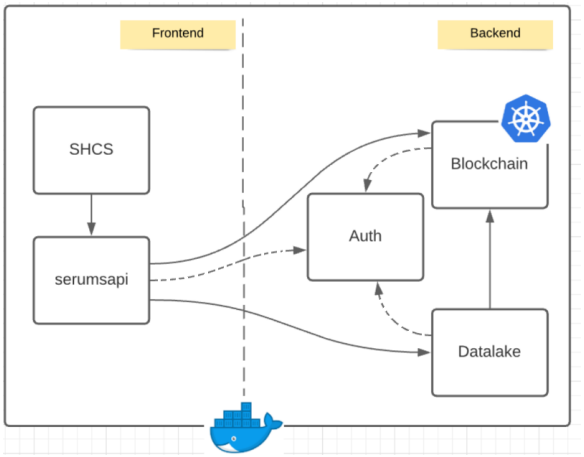


Figure 5 Serums components interaction

5.4 Deployment Viewpoint

The Serums tool-chain [1] is composed of different components or subsystems that can live independently. However, for the SHCS to function properly, all the components must be able to communicate amongst them. We have leveraged the features of the Docker[2] technology to produce a seamless integration (**Figure 6**); this way all subsystems (components) will be containerized and deployed in a single managed server. With this architecture structure, all communications between components happen internally across the same network. The use of Docker containers is common in projects where replication is crucial, such as the case of Serums. All the technical partners can work independently in their modules using different platforms, programming languages, operative systems, etc., and the integration module will be able to interact with them in a homogenized way.

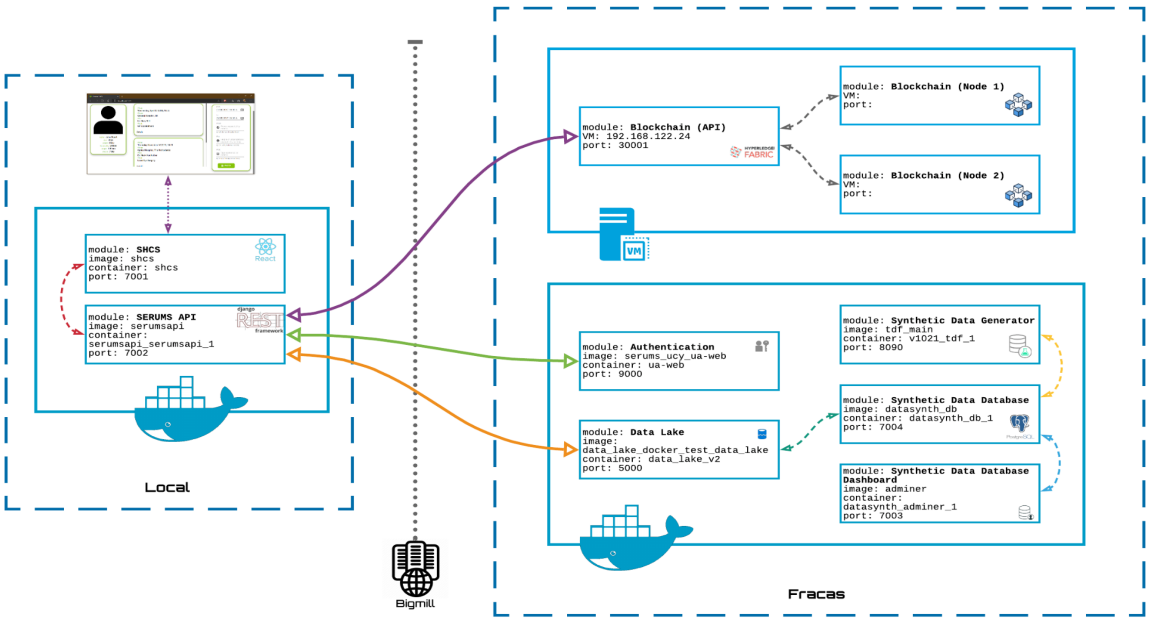


Figure 6 Deployment Viewpoint

[2] Docker technology information can be found at https://www.docker.com/

5.5 Sequence Diagrams on the Core Functionalities

The Sequence Diagram (**Figure 7)** shows the steps required e.g., for a healthcare professional user to retrieve the SPHR of a given patient. We assume that both the patient and the professional (e.g., a doctor) are already registered in the system (step 1, for each user type), but the doctor does not have access rules created for the patient's SPHR yet. Then the first step of the use case requires the patient to login into the system (step 1 - Login patient) and then patients can create access rules to allow professionals to retrieve their SPHR (step 2 - Create access rule). At last a professional requests a patient's SPHR retrieval (step 3 - Request SPHR).

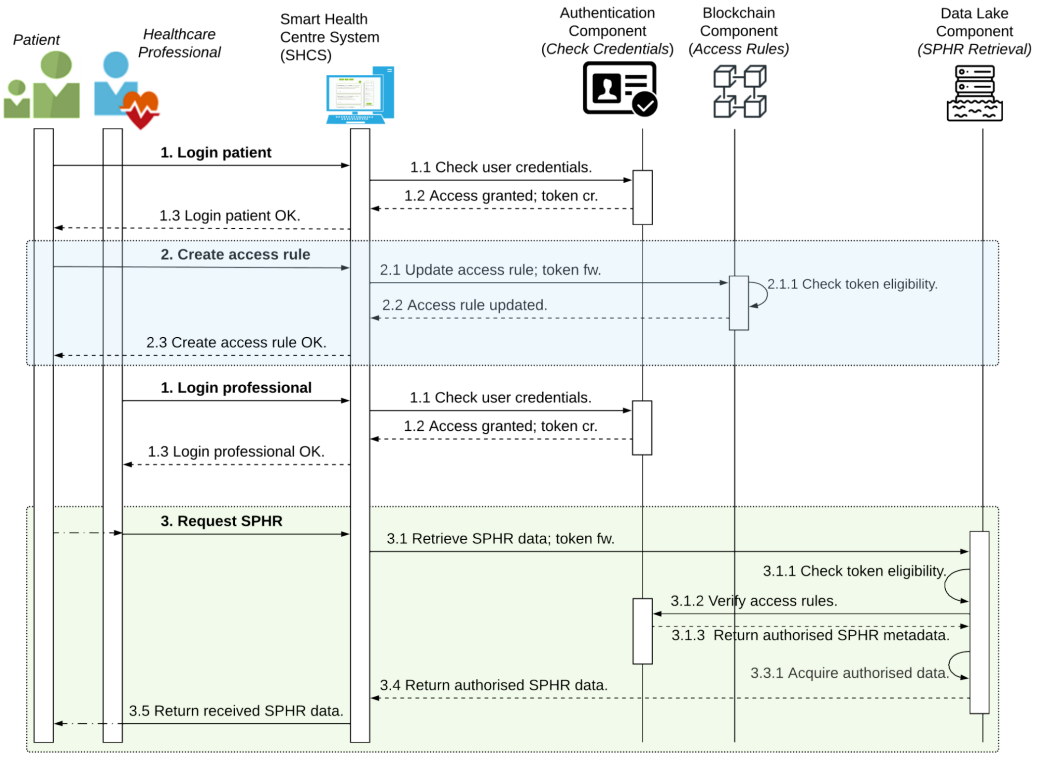


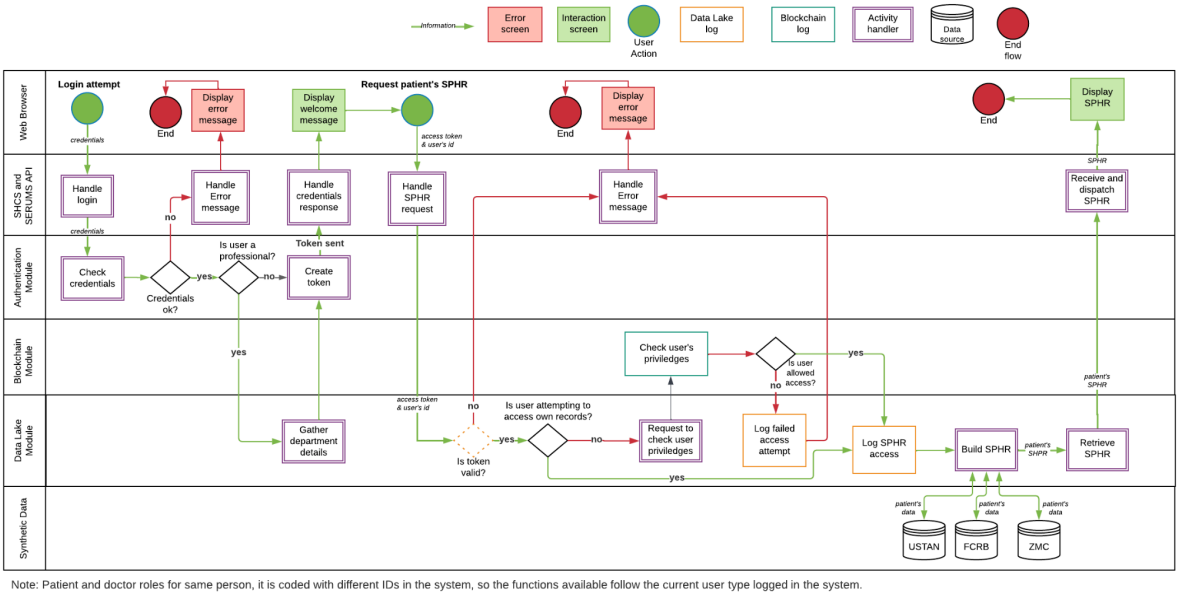
Figure 7 Sequence Diagram (Login and Request SPHR)

In practice, once the patient has logged in from the Authentication client (step 1), they will be redirected back to the SHCS with the JWT token to access all other functionalities (step 1.2). The creation of access rules is an option for both patients and professionals (step 2) in the current version. Once an access rule creation request is issued from the front-end, the system checks if the rule is conflicted considering already stored rules in the Blockchain; if yes, the system requests the patient to amend the rule; if there is no conflicting rule, the system triggers an API call to the Blockchain to register/store the rule. Professionals perform login as well (step 1 - Login professional). Note that the authentication of different types of users follows the same steps; if successful, the authentication system returns an access token (step 1.2) and the front-end enables the menu with options for the user. After login, the professional can then request the retrieval of a patient's SPHR (step 3 - Request SPHR). This request will go through to the Data Lake to retrieve the SPHR information together with the Authentication token (step 3.1). After checking token eligibility (step 3.1.1), Data Lake requests to the Blockchain component to check the user's authorisation detail (step 3.1.2 and 3.1.3), and to the Data Lake to retrieve the SPHR information with the specific synthetic data (3.3.1) this particular doctor is authorised to visualise (step 3.4 and 3.5).

In the PoC3 version, patients as users are able to retrieve their own SPHR, create access rules to professionals, and accept (or reject) new access rules created by professionals to them. Professionals (e.g. doctors) as users are able to request a patient's SPHR and also request new access rules directly to patients. This latest addition contributes to the testing of the Data Lake retrieval input/output combined with the Blockchain authorisation mechanism.

5.6 Information Flow Viewpoint

The Information Flow Viewpoint (**Figure 8**, also shown in larger resolution on **Appendix I**) shows what information flows between each component along the *Use Case* presented in **Figure 7** (which depicts a sequence diagram from the perspective of a user accessing the system).



Note: Patient and doctor roles for same person, it is coded with different IDs in the system, so the functions available follow the current user type logged in the system.

Figure 8 Information Flow Viewpoint on the SPHR retrieval from Data Lake

First, the SHCS requests an access token to the Authentication system using the user credentials; if authenticated, this token enables the user to continue with the navigation in the system. Then the user (e.g., a healthcare professional, a doctor) can request a patient's SPHR to the Data Lake component, where this request will have to be authorised by the Blockchain component, which issues an authorisation response containing the authorised metadata for the user, i.e., it filters the patient's information (categories and subcategories of medical data) to be displayed to the user in the front-end.

In this D6.3 deliverable we include an additional information flow viewpoint, expressing the access rule creation/update feature, considering the patient perspective (**Figure 9**, also shown in larger resolution on **Appendix I**).

Figure 9 shows the information flow concerning the SHCS and the Blockchain component to get an access rule registered. The login attempt process remains the same till the rule creation functionality is triggered by the user, then the request to create a rule is checked by the SHCS for conflicts with existing stored rules. After this process, Blockchain is triggered, first checking the access token and then registering/storing the rule update.

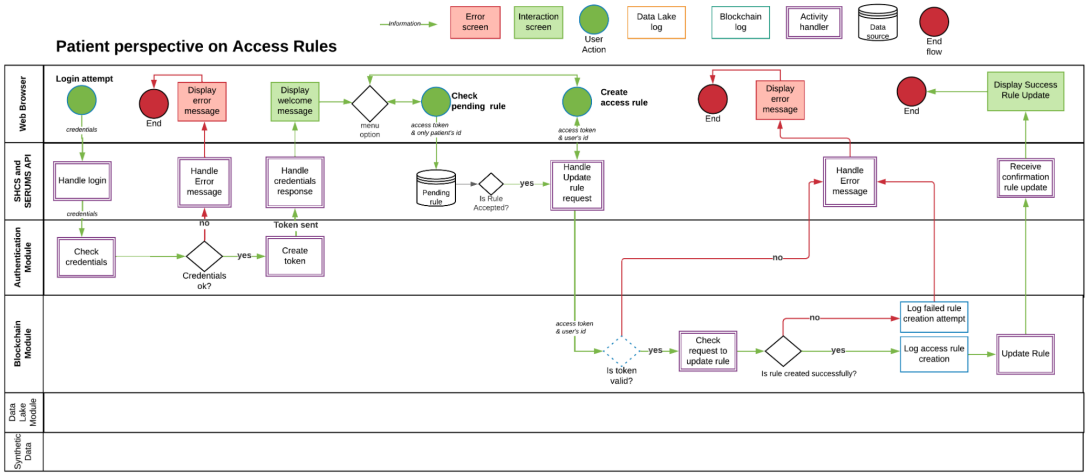


Figure 9 Information Flow Viewpoint on Access Rules feature

These above-mentioned viewpoints (**Figure 8** and **Figure 9**) are the basis of the Serums platform architecture. In the following sections, we present the detailed information on the Serums platform integration as well as detail on the front-end - the Web User Interface (WUI) integration.

6 Serums Integrated System: integration phase

In this section we give some details on the implementation of the system. The SHCS is a client-server application where the front-end (shown in the **Figure 10** below as *Client(shcs)*) is written in JavaScript and developed with the React framework as is current practice for the development of such systems.

The system integrates the following components:

- the ***Authentication system*** (WP5), consisting of a client (Auth client) and corresponding backend (Auth module) which performs the actual authentication. This component is further detailed in WP5 and associated deliverables.
- the ***Data Lake component*** (Data Lake in **Figure 10**) which contains and manages the Smart Patient Health Record (SPHR) (WP2).
- the ***Blockchain component*** storing the access rules (WP2).
- the ***Questionnaire component*** for system end-user evaluation, which was a joint development between WP2 (coding of the questionnaire access and storage), WP6 (concerning facilitating the access to the questionnaires and linking them to the remaining system with front-end and integrated components) and WP7 (questionnaire content creation with translations available in Dutch and Catalan languages for the users of the respective countries). This component just exists for the purpose of evaluation and is not an integral part of the system itself.
- the SerumsAPI which is a proxy server (shown in the **Figure 10** as *serumsapi*) to simplify communication to the components and provide additional functionality to guarantee correctness. The functionality provided includes a ***Rule Conflict Detector,*** which filters out conflicting rules to ensure that the registered access rules in the Blockchain component are conflict free; and a ***Pending Rule DB*** which stores all requests to access data for a given patient (usually by medical staff) to be approved by them at a later stage.

- the SHCS client (shown in **Figure 10** as *Client(shcs)*) which displays health data to a user and allows them to perform different functionality such as browse access rules, create new rules, etc. Only for the purposes of the user evaluation, the client gives direct access to a questionnaire as described above.

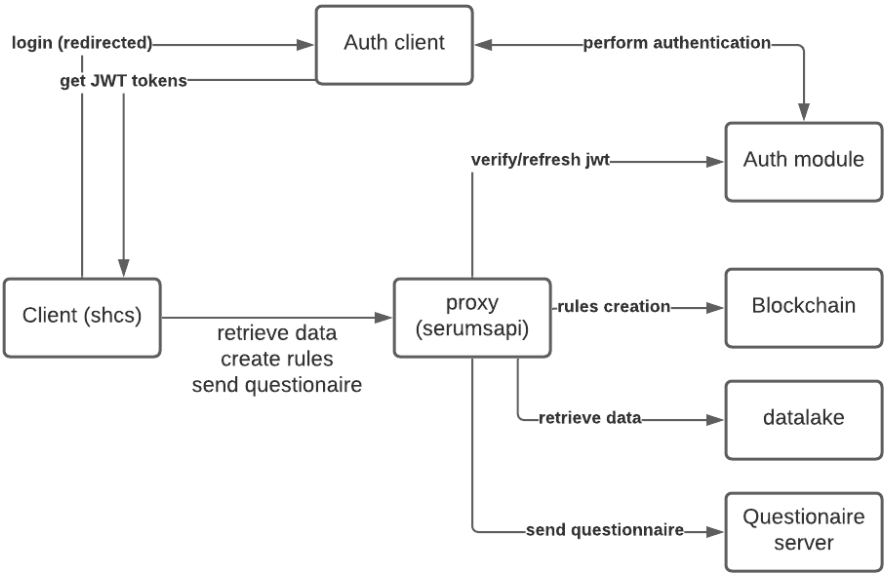


Figure 10 Serums components

It is worth mentioning that we have a language component to translate the page into several different languages. We forward this component as React properties to each module. When a language is selected, the language component provides the correct translation. This was particularly relevant for the user evaluations done in different countries using the questionnaire.

Finally, there is currently no proxy implemented between the Serums API and the IBM data fabrication tool used in WP4 to create the synthetic data for our medical use cases and used in our data lake and for the data analytics conducted in WP3. This will be another feature that could be implemented in future versions.

In the following sections, we describe how the individual components from other WPs are integrated with the client.

6.1 Authentication System integration

The first action of the Serums web system is to redirect users (e.g., patients, professionals) to the authentication web-page[3] *Flexpass* system [8] (WP5). In order to do that, the system implements an *AppWrapper* in its initialisation, a wrapping module to handle the authentication as follows:

1. At first, *AppWrapper* redirects the user to the Authentication client;
2. Once the Authentication client redirects back with JWT (tokens) in the URL, the *AppWrapper* will parse the token and perform verification or refresh token.
3. If the token is verified as 'valid', the *AppWrapper* redirects the user to the welcome page.

We use JSON Web Tokens[4] (JWT), which is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON

[3] FlexPass system: https://flexpass.Serums.cs.st-andrews.ac.uk/web_app/index.html
[4] Information about JSON Web tokens can be found at https://jwt.io/introduction/

object. The information can be verified and trusted because it is digitally signed. The most common scenario for using JWT is on authorisation processes. JWT tokens are created by the authentication module in the Serums system. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token.

JWT created by the Authentication system in the Serums platform retrieves the patient information as a JSON file such as the code excerpt shown in **Figure 11**.


```
{
  "token_type": "access",
  "exp": 1606300849,
  "jti": "2093cabf567f4e2c883d13df07e5b3cc",
  "userID": 56,
  "iss": "SerumsAuthentication",
  "iat": 1606299049,
  "sub": "p1@zmc.com",
  "groupIDs": [
    "PATIENT"
  ],
  "orgID": "ZMC",
  "aud": "https://urldefense.proofpoint.com/v2/url?u=http-
3A__www.serums.com&d=DwIDaQ&c=eIGjsITfXP_y-
DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=uTfN5uQ1khwbRy_TgKH6aUd0-
Bbm0G8K-VajkzZmy98&
m=2iUNn29FSaf7-03xu9xMBrcn4t6U_3w3uqLiLytTfT4&
s=5jB2jmqhsNA_g1SVyZgUFRF9oEP8_AQa-licYW3Iufw&e="
}
```


Figure 11 JSON Web tokens implementation code excerpt

There are two different token types: *JWT access* and *JWT refresh*. *JWT access* (**Figure 11**) is used for retrieving the data from several components (e.g., Data Lake, Blockchain) and *JWT Refresh* is used when the token expires.

A compact JWT structure consists of three parts: header, payload and signature. **Figure 11** shows the payload, which contains the claims. Claims are statements, e.g., about the user, and additional data. The predefined claims are: iss (issuer), exp (expiration time), sub (subject), aud (audience) and others. We give details to some below:

- *exp*: states the expiration date of the tokens in a timestamp format;
- *userId*: is the Serums user ID;
- *sub*: denotes an id in this case the email used for login using the authentication model;
- *orgId*: captures the base organisation for each patient. This information defines the default language for the Serums SHCS client;
- *groupIDs*: defines the user identities. After the user finishes its authentication, the user is redirected to the specific page based on its groupID. On this system version, we implemented several procedures for users that identify as PATIENT, PROFESSIONAL or ADMINISTRATOR.

6.2 SPHR/Data Lake component integration

The Smart Patient Health Records (SPHR) component (WP2) retrieves patient records from the Data Lake, integrating them for displaying the data in the SHCS client. The components within the

SHCS client have been implemented using React Hook[5], which has the benefit of supporting functional testing and avoids managing state, and facilitates independent testing of front-end elements.

Following we describe the interaction between the SHCS client and the Data Lake:

- Data fetcher: fetches the data from the Data Lake. This uses *axios library*, a JavaScript open source library to perform the HTTP request.
- SHCS Client performs GET HTTP requests to the Data Lake via the Serums API. By having the SCHS client routing all access via the Serums API, we avoid CORS (*Cross-origin resource sharing*) issues.
- From the Data Lake, the SHCS client receives the following information: basic patient profile (i.e., name, nationality, height, weight), data tags (previously assigned in the Data Lake processes/WP2), registered rules, and the patient data.
- The SHCS client renders the data grouped into:
  - Basic Profile information showing generic patient data
  - Patient events including appointments, etc.
- In the parent component - 'Health Record' component - for the components mentioned above, SHCS parses the data from the Data Lake module and renders the information accordingly.

Observations:

- We do not store authentication data on the user client for the POC, such as cookies for the jwt access token and refresh. We attached the access token as part of the url.
- We performed integration testing for all the modules; during the testing we found several issues caused by other modules (for example) the Data Lake, such as missing tags, unregistered medical staff and patients. We informed the Data Lake team to make changes and re-run the test.

6.3 Blockchain component integration

The Blockchain component (WP2) is responsible for storing the access rules which are then interpreted by the Data Lake component to control retrieval of any SPHR. The SHCS client also retrieves the access rules and displays them to the patient allowing them to decide whether they are appropriate or not (**Figure 12)**.

Accordingly, the SHCS client has functions to edit, delete and create access rules combined in a so-called *Rule Component*. This Rule Component contains different methods and APIs as follows:

a. If the users (patients) have existing rules created in the Blockchain, the existing rules will be displayed in the Rules page that contains a button 'Create New Rule' and displays the list of 'Rule cards'.
b. A 'Rule card' contains the information regarding an access rule, the one that has been created, an 'edit' button and a 'remove' button.
c. The 'edit' button triggers the *EditRule* method, where the user (patient) will be redirected to the RuleCreation page.
d. The 'remove' button will mark the access rule as deleted in the blockchain and the rule will then be removed from the 'Rule cards' list.

More examples of screenshots involving rules are shown in **Appendix II. D**.

[5] More information about React Hooks can be found at https://reactjs.org/docs/hooks-intro.html

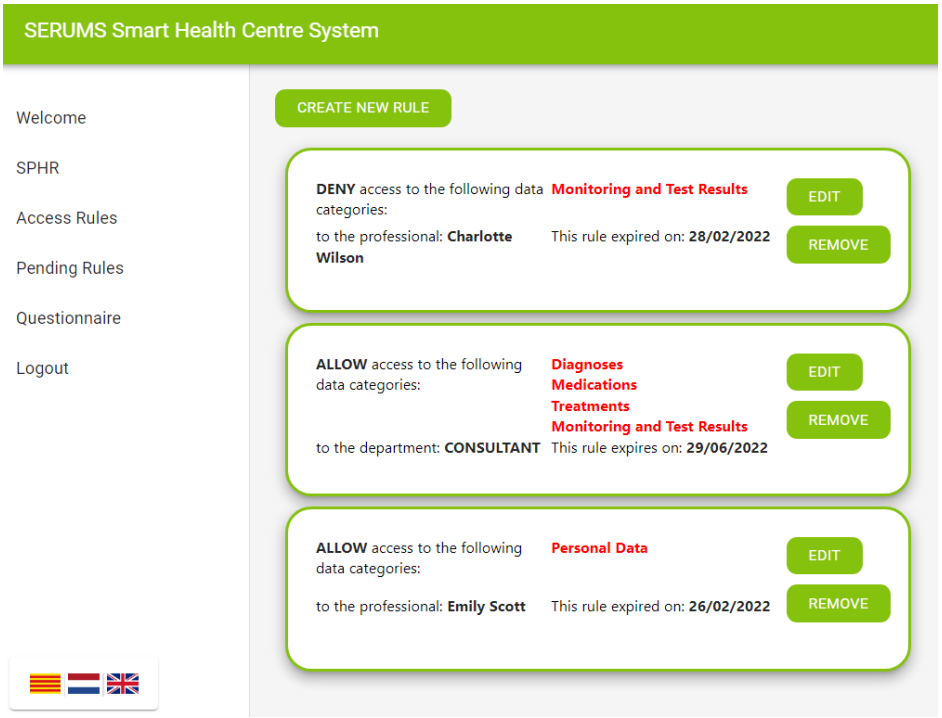


Figure 12 WUI for browsing Access Rules showing three separate rule cards

Overall, how access rules are managed by the SHCS client is directly aligned with the system's high-level requirement establishing that citizens (i.e., patients) must have control of what professionals and organisations have on them in terms of data. From an implementation perspective, access rules are dealt by the SHCS client as follows:

1. When the 'Submit Rule' button is clicked, the SHCS client performs the POST request to the Blockchain component via the Serums API.
2. If the rule is successfully created, the SHCS client performs another POST request to the Blockchain component and registers the access rule there.
3. Because we are required to use different tokens for creating a specific access rule, the SHCS client provides the *refresh token* in the body of the request. With the refresh token, the Serums API performs a POST HTTP request to the Authentication system for getting a new JWT (access token). The new token is used as one of the HTTP headers (i.e., authorisation) for issuing another POST request to the Blockchain module.
4. Once the access rule is created, SHCS fetches the created rules on the main page of the Rule component.

In order to illustrate what happens with some of the functionality involved, we describe the implementation of the RuleCreationPage:

- CreateNewRule:
    - the user will be directed to a new page (RuleCreationPage) to create a new rule for a healthcare professional.
    - the RuleCreationPage contains several properties of an access rule for the users to determine, such as:
        - Action (Allow/Deny);
        - Select healthcare professional or department (in a given selected location);

        - Select the data categories or subcategories (tags);
        - Select the validity (an expiration date for the rule).
- SubmitRule: ‘Submit the rule’/ ‘Edit the rule’ to the Blockchain and the Data Lake.
    - Cancel button: triggers back to the RulePage.

An example of a final POST request body for a new rule is shown in **Figure 13**.

Figure 13 POST request body code excerpt

From the point of view of a healthcare professional, they can request the creation of a new rule for a patient in order to give that professional access to some of their medical data. Such a request is stored as a pending rule. When the patient logs into the system, they can choose to accept or reject the requested rules in the ‘Pending Rules’ option available to them. If a pending rule is accepted, it has immediate effect in the SPHR retrieval for this professional to access the data for this patient.

The underlying schema for this implementation has a formal language and specification that has been developed by researchers in the WP6 team and published as a research paper [4], which brings a formal approach to validate and further check conflicts within the access rules set created by a patient user. This feature is integrated in the SHCS version as a way to guarantee that no data leaks or privacy breaches could occur during the processing of stored *vs.* new rules created in real-time, which have immediate effect in the users profiles.

We present a brief explanation on the formal definition of an access rule as a tuple of information to authorise SPHR data retrieval.

NewRule = (granter (id,type); action; grantee (id,type,hospital); time; tags)

- **Granters** : e.g., patient (data subject); hospital (data controller).
- **Actions** : allowing; denying.
- **Grantees** (data recipients) : e.g., a doctor, a nurse from a hospital.
- **Time** (validity): access rule’s start (creation) date and access rule’s end date; forever; never.

- **Tags** (data categories or subcategories): e.g., consultation, treatment, diagnostic, device, medication, personal information, chemotherapy, comorbidities, hospitalisation, all (data).

Patients are entitled to decide which part(s) of his/her own medical records will have access privileges for a given selected professional/user. The tags (data categories and subcategories) are predetermined by the healthcare organisations (i.e., Serums Use Cases/WP7) as they provide heterogenous types and amount of data concerning their patients' records. In the Serums case, the information about patients is completely synthetic only used for PoC and system evaluations and it is provided by the IBM partner (WP3) - the Data fabrication component [6], and stored as metadata in the Data Lake component (WP2) ruled by authorisation mechanisms on the Blockchain component [7]. Our proposed formal framework operates underneath the WUI enabling patients and other authorised users to securely manipulate the access rules in natural language. Following some access rules examples:

```
rule1 = ((p1; patient); allowing; (d1, doctor); (t1; t2); treatment);
rule2 = ((p1; patient); allowing; (d1, doctor); (t1; t2); personal information);
rule3 = ((p1; patient); denying; (d2, doctor); (t3; t4); (diagnostic, all));
```

About conflicting rules, after creation, we point out that we do not actually have to remove any rules from the set of rules to deal with conflict automatically in the future. A possible approach in the future is to associate priorities as an integer to access rules, and use an SMT solver [9] to always find the set of rules with the highest priority. The current system version checks 'action' parameter, where a DENY rule takes priority over an ALLOW rule as well as Individual rules take priority over organisational/department rules. For example: suppose a given patient 'p1' allows a department from an organisation to access a set of categories of medical data, and doctor 'd1' belongs to that department. Shortly after, 'p1' decides to deny 'd1' access to categories of medical data, ensuring that 'd1' is unable to retrieve and view this patient 'p1' data until conflict free access rules are registered for this particular professional.

6.4 Integrated System deployment

- The Serums SHCS is deployed on the Fracas server at USTAN. It is deployed as a docker container at port 7001.
- By using SSL certificate and key, we open this client application to the public domain: https://shcs.Serums .cs.st-andrews.ac.uk.

Serums API summary:

- Serums API is a proxy server that connects the client SHCS and the other modules for Serums application. The application is written in Python using Django Rest framework[6].
- The Serums API creation bypasses the CORS issues. All the components are basically forwarding the request from the SHCS, except when the user creates an access rule. Following an example of the implemented forward function (**Figure 14**).
- Since we used different tokens for creating a rule, we provide the Refresh token to the body. With the Refresh token, the Serums API performs a POST HTTP request to the Authentication system to obtain a new JWT (Access token). The new token then is used

[6] More information about Django Rest framework can be found at https://www.django-rest-framework.org/

within the HTTP header (i.e., authorisation) for making a POST request to the Blockchain component.

- We use both functions and class implementation for more flexibility regarding the shape of our API to implement the proxy with support of the Django Rest framework library as mentioned above.
- Each request from the SHCS requires a JWT (tokens) within the Authorization header.

```
@api_view(["POST"])
def refresh_jwt(request):
    """ Refresh the jwt token

        Parameters: {
            "refresh": <jwt_refresh_token>
        }

        Return: new jwt tokens
    """
    try:
        res = requests.post(REFRESH_URL, data=json.dumps(request.data), headers=get_header(request.META[HTTP_AUTH]),
                            verify=False)
        return Response(res.json(), status=res.status_code)

    except ConnectionError as e:
        return e.response
```

Figure 14 SHCS forward function code excerpt

The **Serums API endpoints** are the following:

- Authentication system:

  We perform the HTTP request to `https://ua-web/ua`, which is the address for the authentication system container.

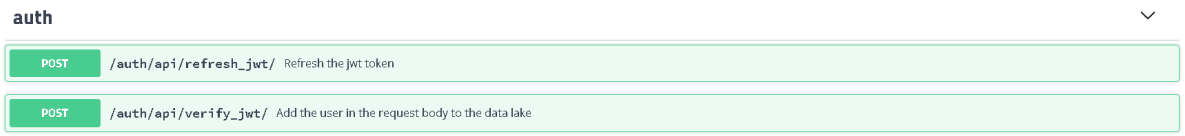


Figure 15 Authentication endpoints detail

- Data Lake component:

  Address/container name in Fracas server: `http://data_lake_v3:5000/`

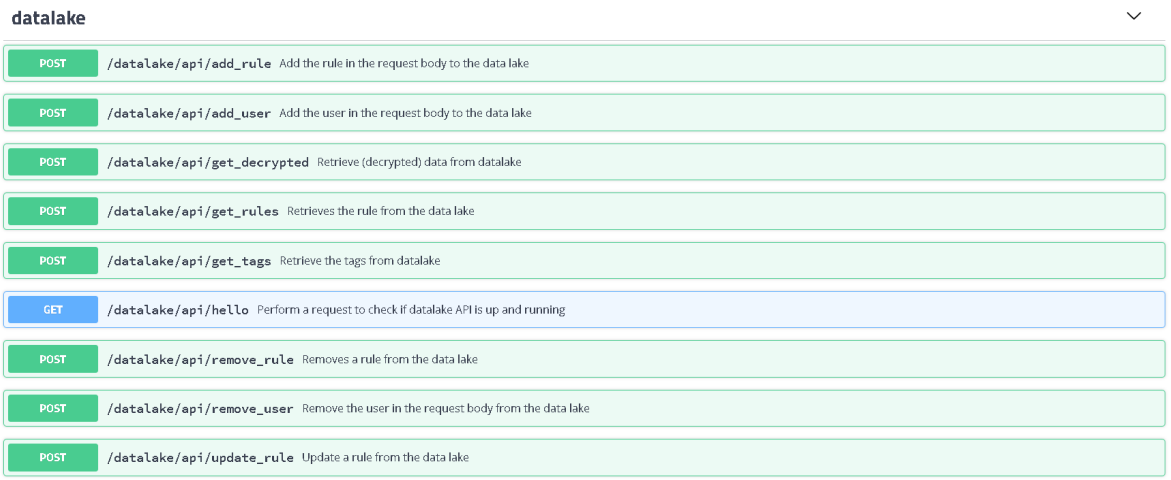


Figure 16 Data Lake endpoints detail

- Blockchain component:

  Unlike the other Serums partners' modules, the Blockchain component is deployed on a Kubernetes cluster on `http://192.168.122.24:30001/v1`

Figure 17 Blockchain endpoints detail

6.5 Integrated System testing

The Serums platform underwent three types of testing approaches for the PoC3 system integration based on the V-model [10]: *unit testing*, *integration testing*, and *load testing*, for both modules - the SHCS client and the Serums API.

The tests for the SHCS client mainly focus on the rendering capability of the client application. The tests for the Serums API focus on the integration between the API and each component (i.e., Authentication, Data Lake, Blockchain, and Questionnaire).

6.5.1 Unit and Integration testing

We tested the capabilities of each system to handle the errors. For these, we have several scenarios, such as: testing valid and invalid request body made from the client application (for Access rules creation and Data retrieval process); testing the validity of data types (e.g., valid/ invalid id); testing request timeout both in the *client* and *proxy* server; testing several metadata errors and their adequate handling; testing for fake JWT data.

For the Integration testing, we use *Cypress*[7] software to mimic the patient behavior when accessing and using the functionality of the SHCS Client. For creating the testing scenarios we applied the PoC participant's script provided by WP7 as the basis, since it contains all the required steps for the user to evaluate the system including navigation and expected behavior.

The core testing scenarios include:

- Patient user accessing own SPHR data;
- Access Rules creation by the patient;
- Healthcare professional user accessing patient data;
- Healthcare professional user requests access rules for the patient;
- Patient user accept/reject the requested access rules by Healthcare professionals users;
- Administrator user registering the patient to one of Serum's hospital partners.

[7] Cypress: https://docs.cypress.io

6.5.2 Load testing

We performed the load testing to determine the system's performance under real-life load conditions. e.g., 500 users accessing the system simultaneously. We only simulated 500 users simultaneously using the system due to the limitation of the server in which SHCS is currently deployed. For the Load testing we used *Gatling*[8] software (using *Scala Lang*). We simulated several scenarios as mentioned below:

- The core load scenarios include the following features:
  - Patient user access SPHR;
  - Patient user Create Rule;
  - Healthcare professional user Request Rule;
  - Healthcare professional user view Patient's SPHR.

6.5.3 User acceptance

During Poc3 execution, we evaluated user acceptance, although we are aware that User Acceptance Testing (UAT) is one of the last stages of the software development life cycle [10]. During PoC tests execution we performed an UAT based end-user testing as a way to collect users' opinions about the system and to detect errors or inconsistencies for further improvements in the whole system. The SHCS, particularly the *Flexpass* system [8] (password creation feature) and the Questionnaire component, both have been tested during PoC3 execution by a group of 32 participants (USTAN location). After Ethics committee approval, recruitment involved creating and distributing an engaging public information flier to describe the project in an accessible way, including a QR code for volunteers to access further information from the project website.

Applicants were interviewed in real time by video link on Teams. This allowed a high degree of confidence in both the participant's legitimacy and locality. The USTAN team applied this 'COVID compliant' series of Teams-based test days, where participants were given access to the Serums platform, supported by Serums team members in online participatory sessions. Participants were scheduled across seven days from the end February to the beginning of March (2022), with at least two members of the USTAN team available at each online interview. Using a pre-agreed participant's script, participants were asked to create their own picture password in the SHCS by choosing an image relevant to them, then performing a series of tasks to test the usability of the system. This was achieved by requiring participants to share their screen with the USTAN interviewers, which allowed help to be offered if required and allowed early identification of any technical issues. Participants were asked to stop screen sharing for the duration of their participant questionnaire, to preserve the confidentiality of their responses. Once they had completed the questionnaire, participants were thanked for their contribution and reminded that they would receive post-study emails requesting them to log into the system on days 1, 3 and 5 (post-study process) using their initial access (created password) to the SHCS. Following at least two successful (or attempted) logins, each participant was emailed a link to activate their £10 Amazon Voucher in retribution for their time testing the SHCS.

After PoC3 execution, involved partners gathered the lessons learned for the latest improvement of the system features. In the PoC2 we have added one important feature to the client application: the error handling and coding refinement (that has been overlooked before running PoC2). During PoC3 the client application also informed the user if a given error happened. A positive outcome of performing the PoC3 before reporting D6.3 is that all SHCS components and

[8] More information on Gatling software can be found at https://gatling.io/

APIs had another round of testing with end-users, providing important feedback from users' perspective which is a major goal of the Serums project - meeting the users expectations.

6.6 Other discussion on the Integrated System dependability properties

In parallel with the development of the Serums platform, in the context of WP6, we have conducted research on the desirable dependability properties [17] for healthcare systems like Serums, especially *security*, *reliability* and *resilience* aspects. Following a summary of WP6 discussions during this phase of project execution, including articles published in scientific conferences [18,19,20].

Within security, we focused mostly on the aspects of confidentiality and integrity, including principles of 'security by design' to be compliant with the GDPR such as the alignment of system components to include defensive measures in several levels. We performed several security quality assessments to gather important requirements for the platform, in a joint effort among Serums teams (WP2/WP6/WP7).

The research work provided by WP6 entails the validation and verification of the overall system mostly to improve the architectural design decisions (please refer to **Section 5**); for example, in using authentication tokens to identified Serums users, the adequate format of the access rules to reach efficient and secure processing within SHCS, and most importantly, to make the Data Lake and Blockchain components' interconnections safer in case of cyber attacks.

We co-designed cross-country high-level use case scenarios of data access (WP6/WP7) to uncover possible security vulnerabilities, including the analysis of conflicts in access rules when registering patients in multi-national organisations, together with the access rules engine automatically checking for conflicts. In future, we envision integrating robust mechanisms for conflict resolution such as those found in SMT (Satisfiability Modulo Theories) solvers [4].

In another practical direction, concerning assessment of security risks, especially after Covid-19 pandemic that increased the number of cyber threats to healthcare organisations through social engineering, we have evaluated a few common threat scenarios (e.g., *Phishing attacks*) to the Serums platform and how the platform would react to these threats, research that was published in scientific conferences [18,19]. Furthermore, we discussed in published papers [3,5,6,8,18,19,20] that Serums has multiple levels of security, reliability and resilience built into the platform, with the combination of Data Lake and Blockchain components providing integrity and availability. The SHCS portal is a limiting factor to availability [18], as all processes must take place through it. However, there is no undue stress placed on the SHCS, as it only translates requests into a format the other tools require (The SERUMS API and front-end). In this manner, the SHCS is no more insecure than numerous other front-end portals used in multiple other web applications [20]. This does not mean that the security of the SHCS should be taken for granted, however, we have performed an analysis of potential attack vectors, and what defenses would be included [19] within the system itself, as well as what other complimentary systems that would be required (NIDS, Firewalls, etc) [18].

6.6.1 Considerations on the benefits and security risks of the Serums platform

This section brings a discussion on the Serums security aspects published in scientific papers during project execution [18,19]. We have developed a threat model [18] and applied other formal modelling approaches (e.g., Attack-Defense Trees and Mal-activity diagrams) [19], that the Serums integrated system can respond to cyber threat scenarios in a unique fashion due to the blend of security mechanisms integrated in the platform. For instance, while password leakage can never be fully eliminated, the login process within the Authentication system using Flexpass and multi-factor

authentication [8], among other combined approaches in the future, can reduce the likelihood of them occurring. As a strategy to mitigate the user's risk, security awareness training may be an option for administration staff as well as security policies for these actors such as login restricted to trusted hardware, which can increase the typical login security.

Access rules are of great benefit in Serums platform for increasing patients' privacy and data confidentiality. In case of malicious use of this feature, and further detection, the system allows easy restoration of the original rules. With notification that rules have been changed, this restoration could happen in a short period of time. It is noticeable that none of the above technologies precludes the need for more standard security measures.

Furthermore, in practical terms, securing systems is an iterative process, meaning the work is never complete, but refinements and improvements to match the changing landscape are being rationalised. Serums is flexible to some extent in this aspect though being modular. As emerging technologies become more prominent and secure, parts of the system can easily incorporate these new technologies. For example, including biometrics in the Authentication system would decrease the likelihood of Phishing attack attempts, as even if successful, the attacker now would need to possess the attribute being measured (e.g., fingerprint, retina, etc), or a method to bypass the check.

Finally, system logs are a critical part of the system security maintenance, forming an essential part of both the detection of malicious activity and forensic analysis in the aftermath. As such they are frequently a target for malicious actors to hide their activity in the system. Consequently, it is also important that the integrity of these logs can be trusted. Serums platform supports this security challenge using the Blockchain, meaning that any transaction can be verified through checking the indelible activity recorded on the chain. Discrepancies can then be identified, giving clues to the nature and extent of malicious actors' activities.

6.6.2 A brief discussion on the design of a trustworthy, reliable and resilient platform

This section brings a brief discussion on the reliability and resilience of Serums architecture design, already published in a scientific paper during project execution [20]. We have predominantly discussed how Serums design attempts to ensure that the platform is resilient to data breaches, leaks, or even corruption, and how parts of the Serums system could go down without impacting other areas, reflecting on its reliability aspects. However, the added features, while aiding resilience overall, would not ensure a resilient system themselves. Typical backups would still be required, as well as well-defined processes to determine how and when to initiate and use those backup procedures.

One of the contributing factors to Serums platform's resilience is that it only manages copies of medical data when SPHR requests are completed. Hence, in case of system failure, hospitals' medical records are not affected. This means that should Serums go down, a server restart is sufficient to make the system up and running. When combined with the modular engineering approach, which means that aspects of Serums platform could fail without impacting other areas, we believe this could result in a system that is unlikely to suffer large scale failures and be more easily recovered in case of system disruption.

There are effectively only two main components that could cease most of the Serums' functionalities should they fail:

- First, the **Web portal**, which would logically hinder users from accessing the system if it were to go down. However, even this would not cause all functionalities to be unavailable, as data is uploaded to the data lake, and any analysis can be performed independently of the web portal as a measure to increase resilience.

- Secondly, the **Blockchain**, which is essential for authorising users' access to medical data, and if it were to fail, then users would therefore not be able to gain the permissions necessary to access the data.

It can be argued that the Blockchain is the most resilient aspect of the Serums system, with it being distributed between multiple medical centres. If one medical centre version were to go offline, then users would be directed to another node. The nodes are not user-centric, and all store a copy of each other's data. When the offline node comes back online, it can copy any changes made while it was offline. In the meantime, the worst case scenario is that access is delayed slightly due to larger network hops to ensure access permissions.

The different robust technologies involved in Serums platform also have different trade-offs, but their advantages can function together to build a holistic system that is resistant to errors and failures. For instance, disadvantages of the Blockchain when it comes to handling Big Data are mitigated by the Data Lake, while its advantages include immutable logs on system access' attempts and respective completed SPHR transactions, which can aid in system recovery.

The Serums platform allows the components to communicate and effectively operate together, depending on functionality required, with no tighter integration that could cause a cascading failure in the event of one component failing. We believe the blend of these technologies brings key attributes to enable a secure, reliable and resilient data sharing platform for healthcare provision, once combining Blockchain to log any activity and allow easy restoration, with a Data Lake which is perfectly suited to large-scale data manipulation and retrieval.

7 Serums Integrated System: verification process

We use formal methods for the verification of the Serums platform. The approach works with a formal representation of both the platform and its requirements. Systems are commonly represented with transition systems that can be seen as graphs where nodes model states of the system and edges or transitions show how the system can change states. The properties are formalised with one of the Temporal Logics that allow reasoning on sequences of transitions.

Statistical Model Checking (SMC) [11,12,13] is one of the formal techniques that is known for its scalability. The core idea is to run a large number of simulations on the formal representation of a system under verification and to use statistical methods to decide the probability of a property being satisfied. Being simulation-based, SMC has low time and memory consumption in comparison with other formal methods and, consequently, can validate larger systems. SMC has been applied in multiple projects, e.g. [14,15].

In parallel to the development of the Serums platform, we are building a formal model of the system. The model includes all components of the Serums system as well as components representing behavior of the users. We are using the Uppaal tool[9] that provides an expressive modelling formalism and a statistical model checker [16]. Models in Uppaal are defined as networks of timed automata.

An example of an automaton is shown in **Figure 18**. An automaton can be considered as a graph where nodes are states of the system (circles on figures, can have cherry colored names) and edges are transitions defining how the system changes states (black arrows between states). Transitions have a set of optional labels:

- A local variable definition (brass labels) to be used in the other labels.
- A guard is a Boolean expression controlling the enablement of the transition (green labels).
- A channel allows automata to synchronize actions (cyan labels). This label contains a special channel variable and either '!' or '?'. Two automata having transitions labeled with the same

[9] More information on Uppaal tool can be found at http://www.uppaal.org/

channel must take these transitions simultaneously. One of the transitions marked with '!' is an initiator of the synchronisation or a sender, another with '?' is a receiver.

- An update is a sequence of actions that modify the variables of the model (dark blue labels). The updates are defined with a subset of C language.

Several timed automata are combined into a network via synchronisations and shared variables. Note that the same automaton (called a 'template' in Uppaal) can be instantiated multiple times in the network: all instances are independent but have identical behavior. For example, to model multiple patients, a single template is created and multiple patient automata are instantiated. At each point of time the network has three options to evolve to the next state: 1) by passing time 2) by one automaton making a transition that is not synchronized with any other automata 3) by automata making a simultaneous transition synchronized over the same channel.

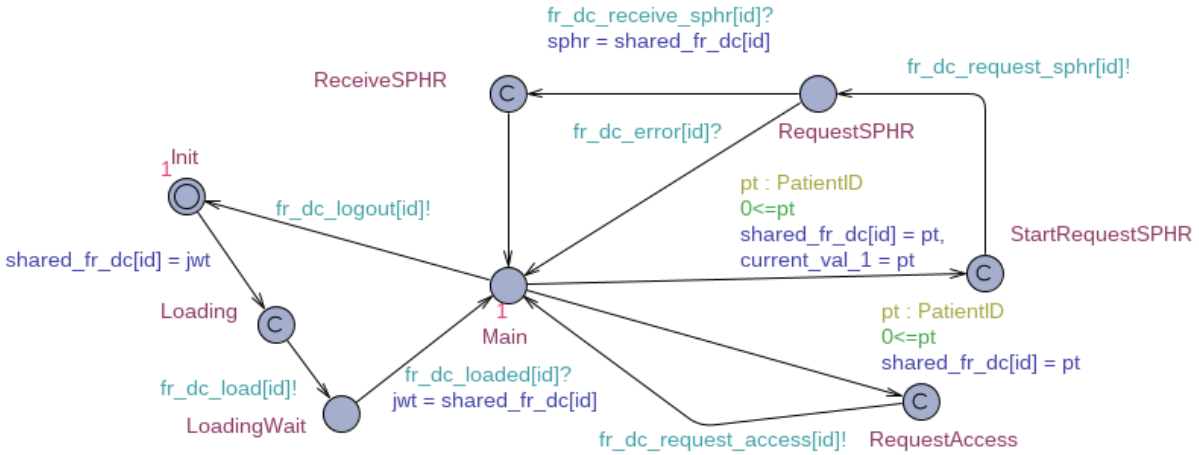


Figure 18 Example automaton

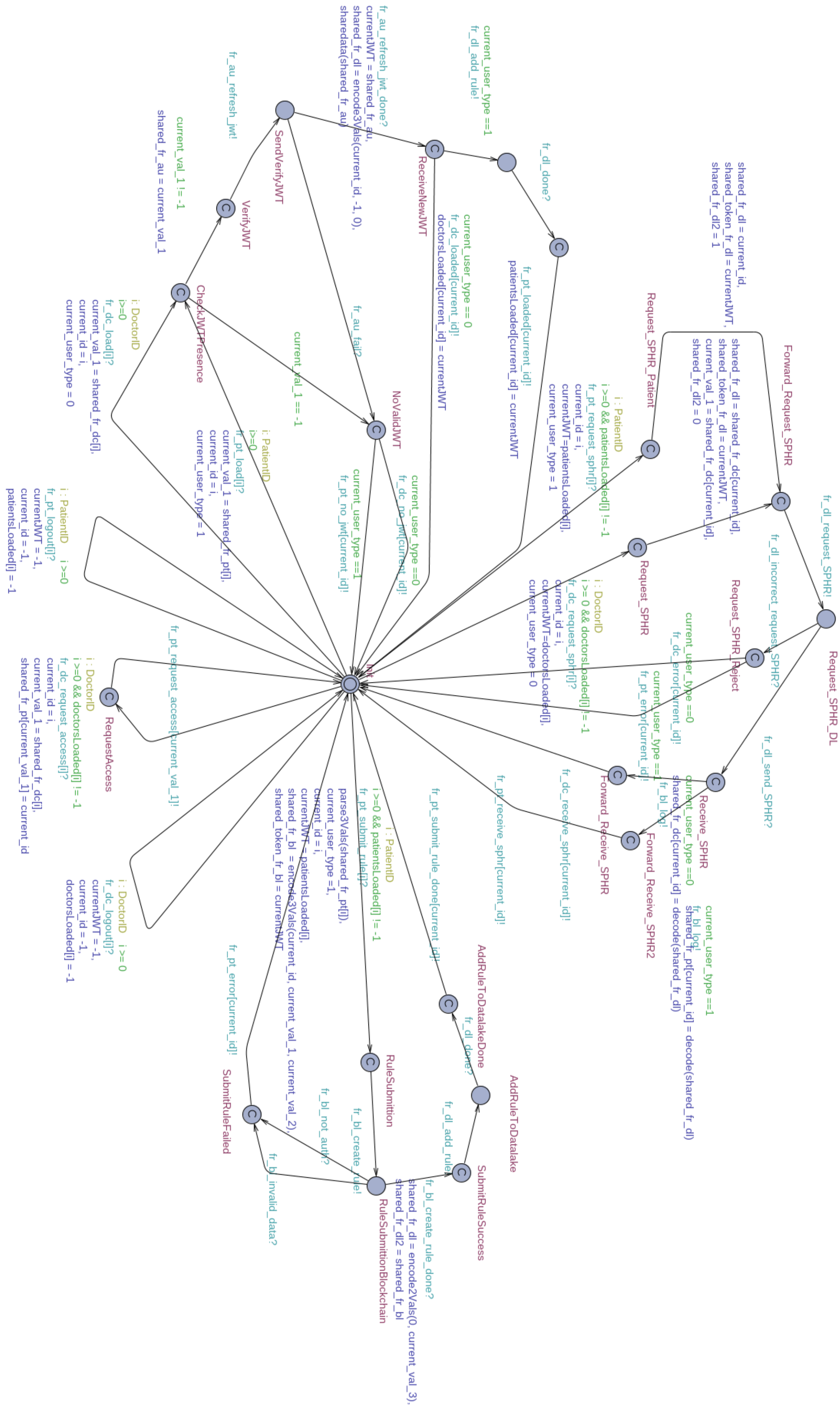


Figure 19 SHCS model

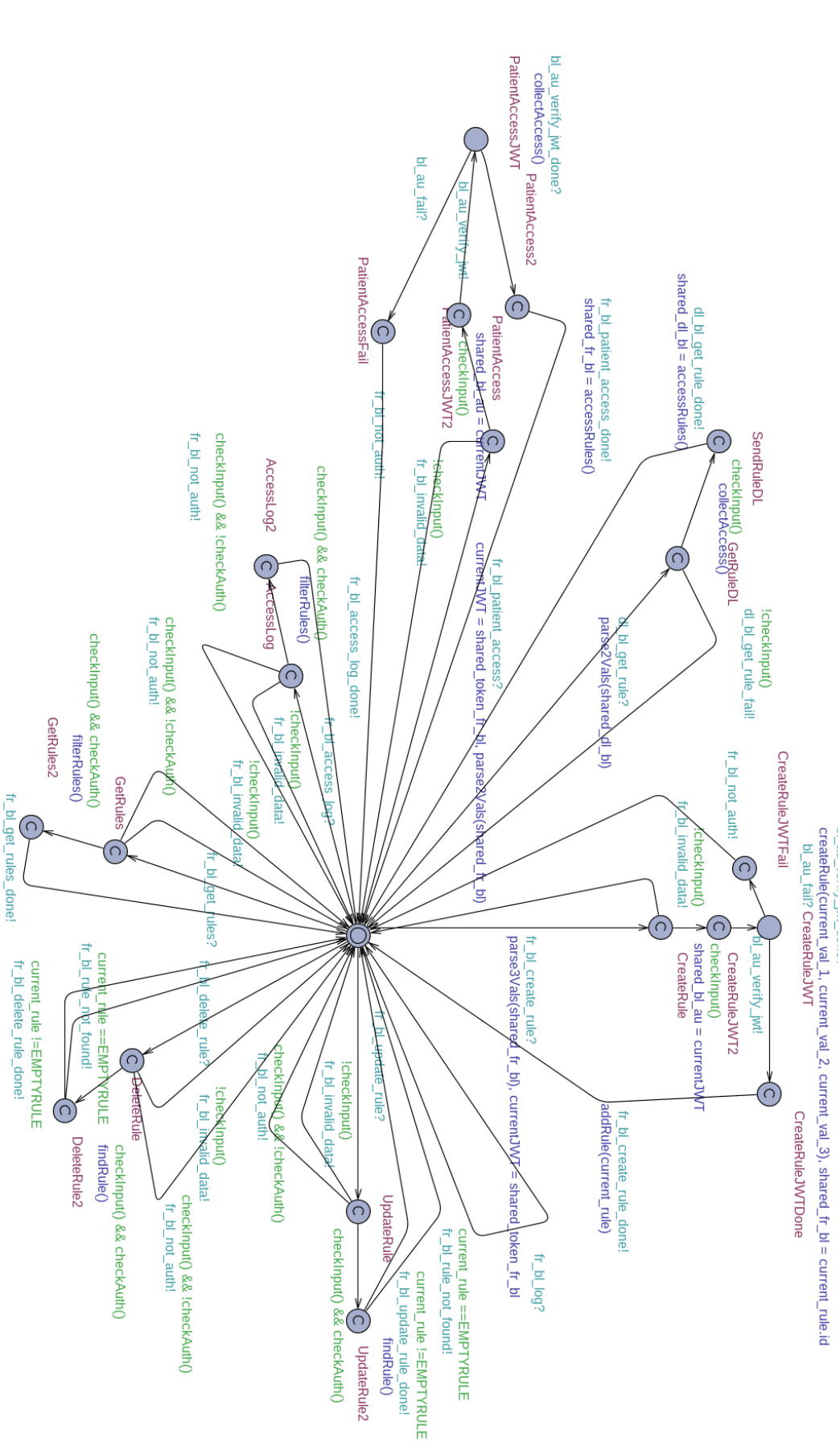


Figure 20 Blockchain model

7.1 Model of the Serums Integrated System

The Serums model is based on the implementation developed within the project. The first version of the model has been based on the implementation for the second Proof-of-Concept (PoC2) and the succeeding versions include modifications done for the PoC3. The model consists of 11 templates: SHCS, Blockchain, 2 template automata for Data Lake, and 4 template automata for Authentication. Three remaining automata describe behaviours of patients, doctors, and administrators of the Authentication component.

Four automata for the Authentication component and its administrators are described in deliverable D5.4. Therefore, we are omitting these 5 automata from the description as well as properties related to Authentication.

SHCS automaton models both SHCS module and Serums API are shown in **Figure 19**. The central state is the initial state of the automaton. The automaton has the following parts:

- Behaviour to the left[10] of the 'Init' state describes actions taken during the connection of a user to the SHCS. If the user has a JWT, the component asks the authentication system to check the token and logs the user in. In case of non-present or invalid JWT, the user is forwarded to the authentication system.
- Creation of rules for the access to patients' data is in the right part of the automaton. A patient is required to be logged in and to have a valid JWT. Information about doctor id, tag, and whether the rule would allow or deny access is transferred from the patient via a shared variable. At the next step the SHCS sends a request to the Blockchain and afterwards to the Data Lake automata. In the current version, the request for an additional access token (Section 6.3) is not included in the automaton.
- A request for an SPHR by a doctor or a patient is shown in the top part of the automaton. At the beginning the Blockchain is requested for the access rules. If the doctor is allowed to receive data about the patient, the corresponding request is sent to the Data Lake. Note that the request to Blockchain is not yet implemented in the Integrated System and the behaviour originates from the Information Flow Viewpoint (**Appendix I.A**).
- A request for access from a doctor to a patient is in the bottom-center part. Due to the fact that listing of all patient's rules is not modelled directly, details of the request are stored in the patient's automaton and the patient can either approve or reject it.
- Remaining transitions in the bottom of the image correspond to log out actions and doctor's request for access.

Figure 20 illustrates the automaton for the Blockchain module. This automaton models the interactive behaviour of the Blockchain module with the rest of the Serums system. It can receive different requests from the SHCS component and (after internal validation) reply back. The requests include creation and modification of rules as well as check for patient's access rules. In the model we abstracted the notions of rules by maintaining the access matrix for doctors and patients. Creation and modification of rules modify the corresponding cells of the matrix.

The Data Lake module is modeled by two automata shown in **Figure 21** and **Figure 22**. The former presents local instances of the Data Lake in hospitals that could be requested for the data during the SPHR creation. The latter one describes the central part of the Data Lake that interacts with the SHCS module. It manages 3 types of requests: add users, remove users, and request SPHR. The former interactions simply update the internal variables, while the last one requests local instances to collect the required data, aggregates the data, encrypts data and sends the encrypted data to the SHCS.

[10] Note that Figure 19 is rotated by 90°, directions in the text are given with respect to the original image.

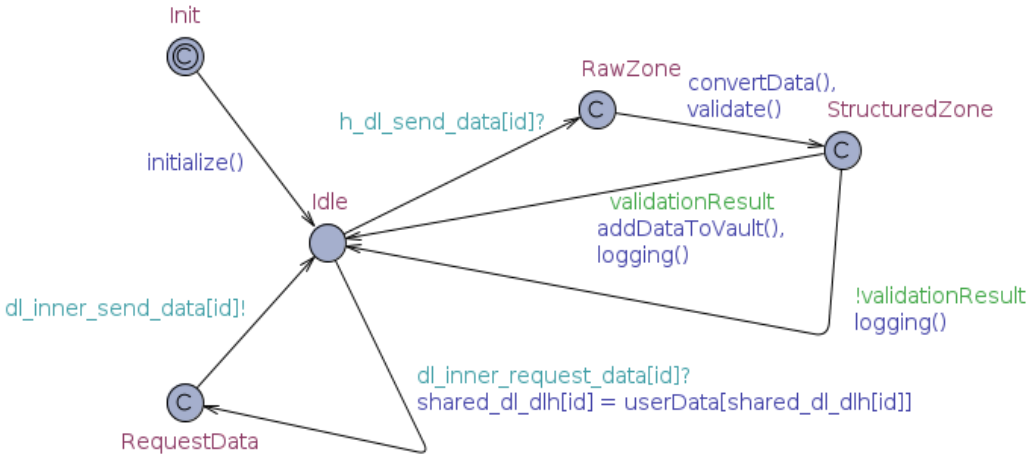


Figure 21 Model of Data Lake local instance

Figure 23 and **Figure 24** show the models of patients and doctors. These two models share a large common part corresponding to the authentication (login and sign up). Both automata start at the bottom-right state and try to connect to SHCS. If there is a valid JWT (have signed up and logged in before) the SHCS would transfer them to the Main state, otherwise to the PGAView. From the latter state they would follow authentication procedures (see deliverable D5.4 for details). After being logged in, patients can create rules for the data access and both users can try to request SPHRs.

7.2 Properties Verification

We use a Statistical Model Checker provided with the *Uppaal* tool. *Uppaal SMC* [14] uses an extension of *Metric Interval Temporal Logic* (MITL). It provides queries to check probability estimation – probability of the property to be satisfied within a given trace length. Basic temporal operators in MITL are *[]p* and *<>p*. The former checks that *p* holds in all states, and the latter checks that *p* holds in at least one future state. The properties has the following format: *Pr[# <= N] F*, where *F* is the property to verify, *N* is the maximal length of traces, and # indicates that we consider the number of transitions rather than time.

Intuitively, the property is checked on a large number of simulations generating traces containing at most *N* transitions and *Uppaal SMC* returns the probability of the property satisfaction with a selected level of confidence. Interpretation of the output depends on a property and, consequently, on a temporal operator used in *F*. For liveness properties where we check that something is possible (e.g. a patient can create a rule), *<>* operator is used and the property is considered satisfied if the probability is not close to 0 and unsatisfied otherwise. For safety properties checking that something is not possible (e.g. receive an SPRH without allow access rule), *[]* operator is used and the property is satisfied if the probability is close to 1 and unsatisfied otherwise.

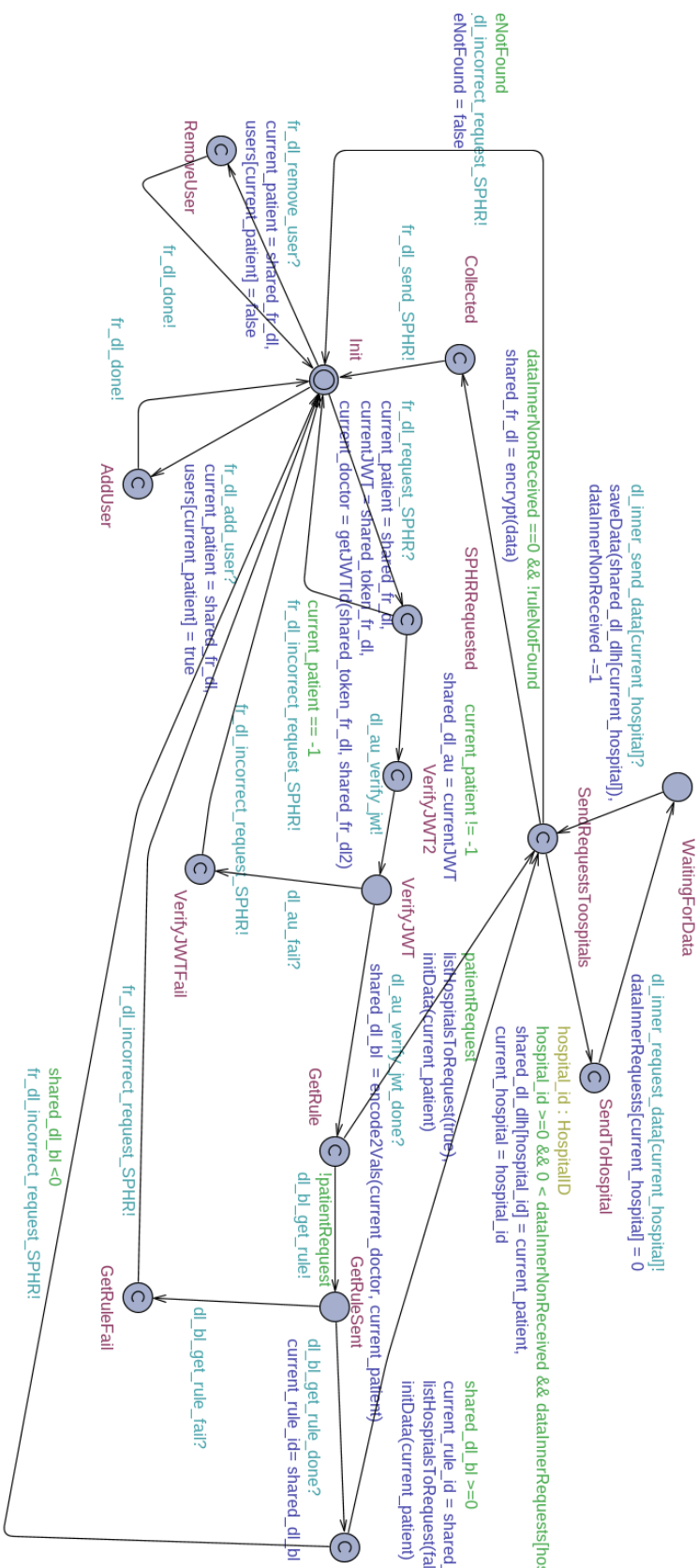


Figure 22 Model of Data Lake global level

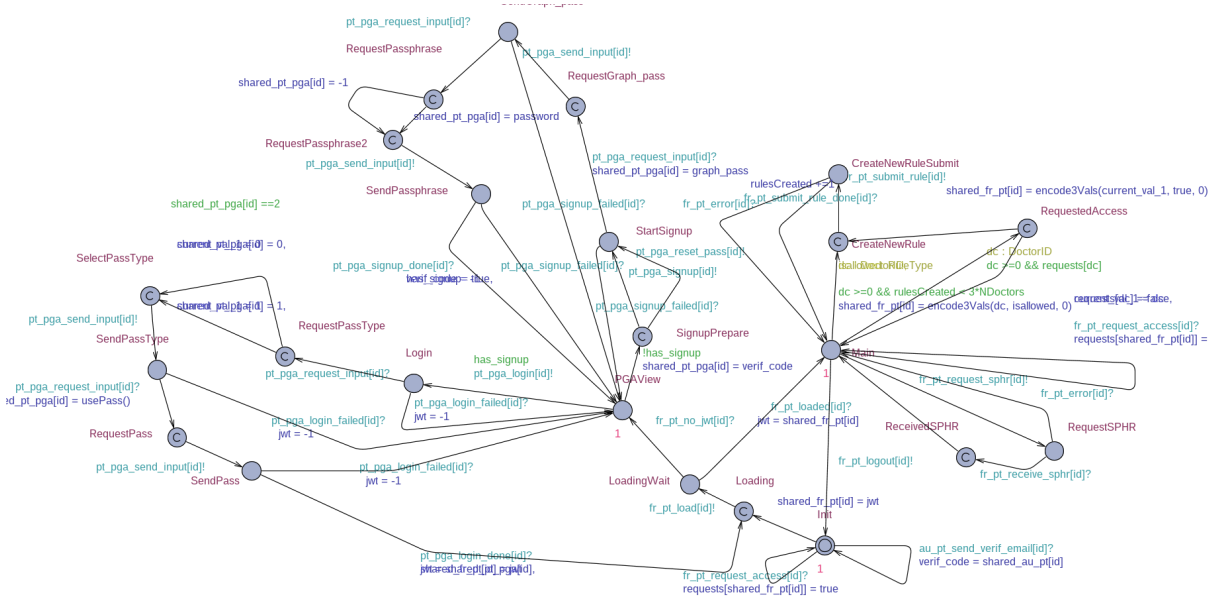


Figure 23 Model of a patient

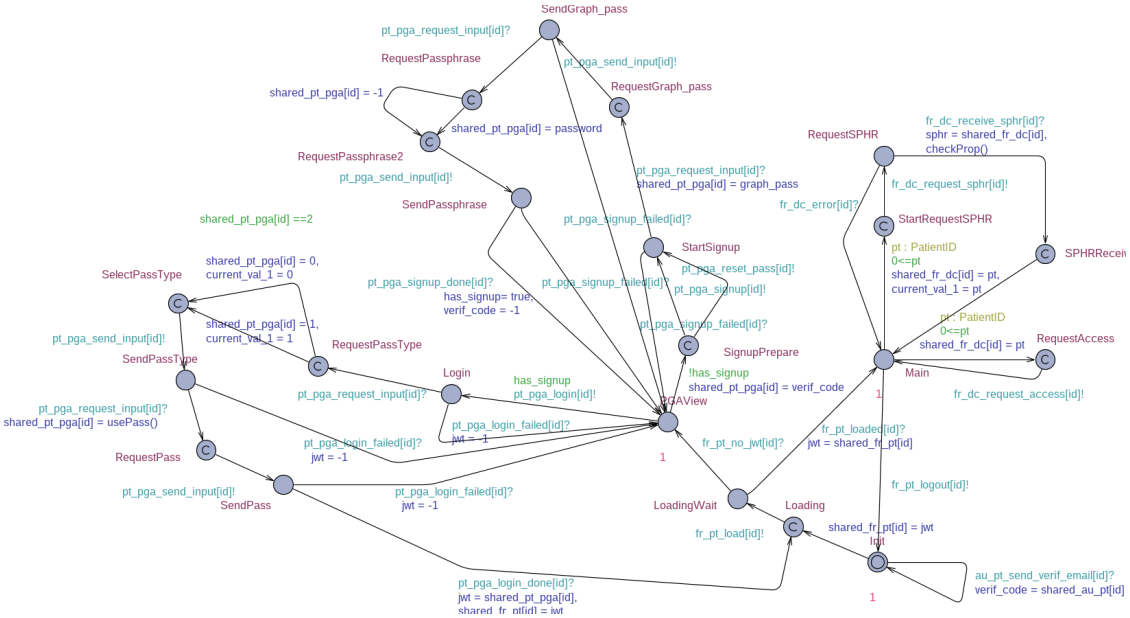


Figure 24 Model of a doctor

In the following table (**Table 1**) we list the properties that have been checked on the model with their description and the corresponding formula. Properties related to the Authentication component are listed in deliverable D5.4. The following notation is used in the formula: -1 represents null or empty value, *d0*, *d1*… are doctor automata, *p0*,*p1*… are patient automata, *bl* and *dl* stand for Blockchain and global Data Lake automata, respectively. Due to *Uppaal* limitation, the properties that every patient or doctor has to satisfy are defined as a set of queries, where each query involves a single user. In **Table 1** we only list the query for *d0* or *p0*, the remaining queries are similar. We use traces of length 20,000, with *Uppaal SMC* it is possible to compute that with this trace length the expected number of requests to the SHCS system is 229.

Table 1 Verified properties on the Serums model

Property	Formula	Notes
No deadlock, i.e. no state where automata cannot progress	Pr[#<=20000] (*[] !deadlock*)	
Medical staff can receive SPHR provided an allow access rule	Pr[#<=20000] (<> (d0.SPHRReceived))	Assuming there is at least one patient that can create allow rules.
Medical staff cannot receive SPHR if access rules deny that	Pr[#<=20000] ([] (!d0.SPHRReceived || (d0.sphr != -1 && bl.rules[d0.sphr /separator -1][0])))	
Medical staff cannot receive SPHR if invalid JWT is provided	Pr[#<=20000] ([] (!d0.SPHRReceived))	Edited model to force sending an incorrect JWT
Patient can view their data	Pr[#<=20000] (<> (p0.SPHRReceived && (shared_fr_pt[0] / (separator -1) == 0)))	
Patient can create and update rules	Pr[#<=20000] (<> (bl.ruleIDs[0][0] >-1))	
Patient cannot create or update rules without a valid JWT	Pr[#<=20000] ([] (bl.ruleIDs[0][0] == -1))	Edited model to force sending an incorrect JWT
Patient cannot receive an SPHR without a valid JWT	Pr[#<=20000] ([] (!p0.SPHRReceived))	Edited model to force sending an incorrect JWT
Medical staff can receive an SPHR after requesting access and being approved by patient	Pr[#<=20000] ([] (!d0.SPHRReceived || (d0.sphr != -1 && bl.rules[d0.sphr /separator -1][0])))	Edited model to forbid patient creating rules that were not requested by medical stuff

Some of the properties have been checked on modified versions of the model where requests from users have been sent directly to Data Lake or Blockchain bypassing SHCS. Model modifications consisted of adding the corresponding synchronised transitions for the request and response. Indeed, for this case, would be to send a request with invalid data: either an incorrect JWT or in case of an SPHR request a faulty identity. On such modified models we checked whether it is possible to receive an SPHR or to create a rule; the verification formulas remained the same.

8 Conclusion

This document is the third and final deliverable of Work Package 6: “Integration and Testing” focusing on the backend and front-end development of the *Smart Health Centre System* (SHCS). Considerable work at this stage of development concerns testing and validation of the integration as is documented in earlier sections, which is essential to ensure correctness and hence guarantee trust by its end users. This deliverable describes the integration of the various technologies into an SHCS system which enabled PoC participants and end-users to perform the different actions in the system. Users were also able to further evaluate it through an online questionnaire, which was included in the WUI after the authentication process was completed, to gain feedback and hence be able to revise different elements of the system as required.

Furthermore, we refined other sections and components for the SHCS to be tailored for professional and admin user profiles. This task ensured that partners from hospitals (ZMC, FCRB and USTAN) provided the WP6 team with WUI requirements for professionals (e.g., doctors, nurses, hospital departments, administrator). Wireframing was a way to design the website service at the structural level. A wireframe was used to lay out content and functionality on a page which takes into account the user needs. The SHCS was also integrated with a complete ‘Access Rules’ feature, coding it with an underlying scheme running in the Blockchain component and directly on the SHCS. The feature includes requirements such as create/retrieve/update/delete operation over rules, and a few other details on parameters (creation date, action, data categories/subcategories visualisation and selection, rule expiration date, user identity profiles and location, etc.).

Concerning the early decisions on the Serums architectural design, it is important to mention that an additional advantage of separating out the SHCS client from the SerumsAPI is that it allows different kinds of clients to be added with minimal architectural changes; for example, a functionally more simple mobile phone application for patients to quickly approve a pending rule request from their doctor or any healthcare professional. Several of these features will be included in the final Roadmap, but their impact on the proposed architectural design is minimal. This also includes further components (or updating components) that may need to be integrated, such as the Machine Learning models and their respective output in a user-friendly visualisation.

The Serums system testing phase included three types of testing approaches (unit testing, integration testing, and load testing) for both modules - the SHCS client and the Serums API. Supporting tools like *Cypress* and *Gatling* software provided the testing environment for developers to focus on the capabilities of the client application, e.g., to deal with errors, as well as on the integration pathways and responses among APIs of each system component (i.e., Authentication, Data Lake, Blockchain, and Questionnaire). Particularly, *Cypress* software enabled WP6 to emulate the patient’s and professional’s behavior in several testing scenarios on user accessing the features of the SHCS Client (e.g., patient accessing SPHR data, creating access rules, professionals accessing patient data and requesting new access rules for patients, and so on). In addition, Load Testing scenarios can provide average estimations on system's performance under practical load conditions. Since Serums platform is a proof-of-concept software, and it is not executing under a production environment, this performance analysis should be further explored in future, considering a different infrastructure and under diverse system settings, i.e., one could scale the Load Testing scenarios on current version, however it would still not be representative of real-world settings. It is recommended that this type of testing be explored in production environment, because it could unveil different performance issues with respect to the overall design and components’ interdependencies. On user acceptance, a positive outcome of performing the PoC3 before reporting D6.3 is that all SHCS components and APIs were

tested with end-users, providing important feedback on their expectations in terms of features and usability (user-friendliness).

The Uppaal models described in this deliverable are refined versions of previous models, which reflect any changes in the code as a consequence of the development of the Serums SHCS. It is important to keep the formal model aligned to the implementation to ensure that any verification result is valuable and reflects the actual implemented system. Safety and security properties were formalised and checked on the refined model again. When checking for security, we added an attacker model to the overall system model and checked the combined model against formulated security properties. We provide different attacker models to describe increasingly more powerful attackers. This use of attack models is useful to explore how it is possible to violate correct system behavior as well as a way to find potential system security vulnerabilities. The modelled properties include: checking functionality 'access' based on user type, unauthorised access to data, replay attacks, man-in-the-middle attacks, logging and log availability.

Finally, the integration of the synthetic medical data, generated with IBM's Data Fabrication Platform, into the Data Lake allowed a complete system evaluation with three types of users and features to compose a trustworthy and secure healthcare platform for data sharing across Europe.

8.1 Final refinement for Months M41-M42

Until the end of the project, we will be working on a very last refined version of the Serums Smart Health Centre System (SHCS) to include a few minor front-end/backend improvements, and evaluate the WUI (front-end) to mostly increase the multi-faceted characteristics of usability (e.g., efficiency of SPHR data retrieval process, effective SPHR visualisation, as well as the integration of a new API for the privacy-preserving data analytics associated with the toxicity predictor of USTAN's use case. Concerning dependability properties such as reliability, resilience and availability, we plan to continue investigating ways to best evaluate/measure these and how to potentially increase them within the platform.

References

[1] Janjic, V., Bowles, J.K.F., Vermeulen, A. F., Silvina, A., Belk, M., Fidas, C., Pitsillides, A., Kumar, M., Rossborry, M., Vinov, M., Given-Wilson, T., Legay, A., Blackledge, E., Arredouani, R., Stylianou, G., Huang, W. (2019). **The SERUMS tool-chain: ensuring security and privacy of medical data in smart patient-centric healthcare systems.** (IEEE Big Data), Los Angeles, December 2019, IEEE Press. doi: 10.1109/BigData47090.2019.9005600

[2] Bowles, J., Mendoza-Santana, J., Webber, T. (2020). **Interacting with next-generation smart patient-centric healthcare systems.** Adaptive and Personalized Privacy and Security Workshop (APPS 2020), UMAP (Adjunct Publication).doi: 10.1145/3386392.3399561

[3] Webber T., Santana J.M., Vermeulen A.F., Bowles J.K.F. (2020). **Designing a Patient-Centric System for Secure Exchanges of Medical Data.** In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12254. Springer, Cham, 598-614. doi: 10.1007%2F978-3-030-58817-5_44

[4] Banton, M., Bowles, J., Silvina, A., Webber, T. (2021). **Conflict-Free Access Rules for Sharing Smart Patient Health Records.** Rules and Reasoning. RuleML+RR 2021. Lecture Notes in Computer Science, vol 12851. Springer, Cham. doi: 10.1007/978-3-030-91167-6_

[5] Bowles, J., Mendoza-Santana, J., Vermeulen, A. F., Webber, T., Blackledge E. (2020). **Integrating Healthcare Data for Enhanced Citizen-Centred Care and Analytics.** EFMI STC 2020. Integrated Citizen centered digital health and social care Citizens as – data producers and service co-creators, Virtual conference 26-27 November 2020, doi: 10.3233/SHTI200686

[6] Bowles, J., Webber, T., Blackledge, E., Vermeulen, A. F. (2021). **A Blockchain-Based Healthcare Platform for Secure Personalised Data Sharing.** 31st Medical Informatics Europe Conference (MIE 2021 EFMI).Volume 281. doi: 10.3233/SHTI210150

[7] Bowles, J., Silvina, A., Bin, E., Vinov, M. (2020). **On defining rules for cancer data fabrication.** RuleML+RR 2020, LNCS, Springer, 2020. doi: 10.1007/978-3-030-57977-7_13

[8] Constantinides, A., Belk, M., Fidas, C., Pitsillides, A. (2020). **Design and Development of a Patient-centric User Authentication System.** Adaptive and Personalized Privacy and Security Workshop (APPS 2020), UMAP (Adjunct Publication).doi: 10.1145/3386392.3399564

[9] de Moura L., Bjørner N. **Z3: An Efficient SMT Solver.** In: Ramakrishnan C.R., Rehof J. (eds) Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2008, LNCS (vol. 4963), Springer, Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-78800-3_24

[10] Mathur, S., Malik, S. **Advancements in the V-model.** International Journal of Computer Applications, vol 1(12), p. 29–34, 2010. DOI: 10.5120/266-425

[11] Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S. **Approximate probabilistic model checking.** In: Proc. of the 5th Int. Conf. on Verification, Model Checking, and Abstract Implementations, Lecture Notes in Computer Science (LNCS), Springer, Cham, vol. 2937, pp. 73–84, 2004. DOI: 10.1007/978-3-540-24622-0_8

[12] Sen, K., Viswanathan, M., Agha, G. **On statistical model checking of stochastic systems.** In: 17th Int. Conf. on Computer Aided Verification, Lecture Notes in Computer Science (LNCS), Springer, Cham, vol. 3576, pp. 266–280, 2005. DOI: 10.1007/11513988_26

[13] Legay, A., Delahaye, B., Bensalem, S. **Statistical model checking: An overview.** In: Int. Conf. on Runtime Verification, pp. 122–135, Springer, 2010. DOI: 10.1007/978-3-642-16612-9_11

[14] Gu, R., Enoiu, E., Seceleanu, C. **TAMAA: UPPAAL-based mission planning for autonomous agents.** In: Proc. of the 35th Annual ACM Symposium on Applied Computing, 2020. DOI: 10.1145/3341105.3374001

[15] Basile, D., Giandomenico, F.D., Gnesi,S. **Statistical Model Checking of an Energy-Saving Cyber-Physical System in the Railway Domain.** In: Proc. of the Symposium on Applied Computing, 2017. DOI: 10.1145/3019612.3019824

[16] David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B. **Uppaal SMC tutorial.** In: Int. Journal on Software Tools for Technology Transfer 17(4), pp. 397–415, 2015. DOI: 10.1007/s10009-014-0361-y

[17] Sommerville, I. **Software Engineering.** Pearson, 10th edn. (2015). ISBN-10, 137035152 (2015): 18.

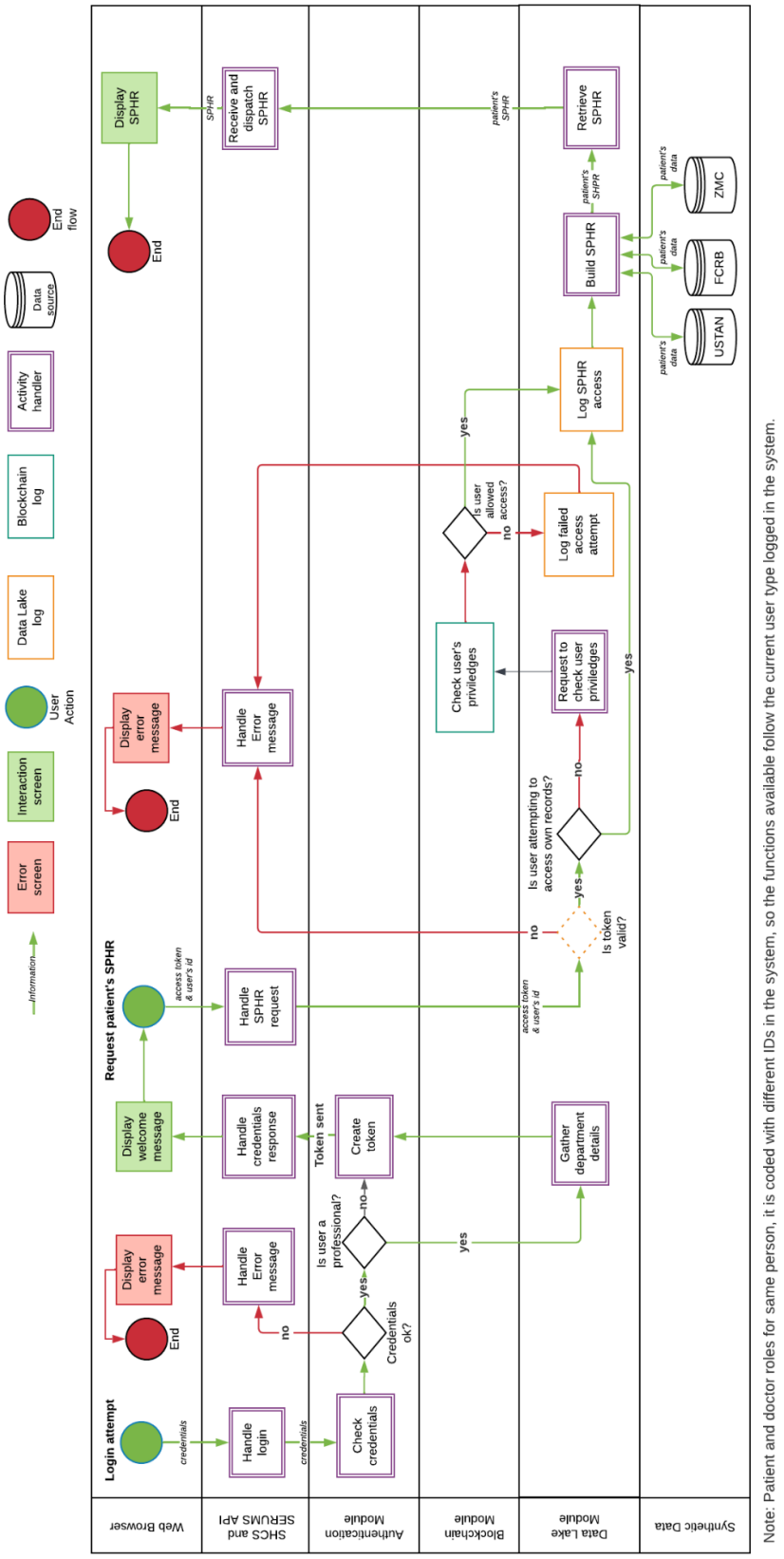
[18] Banton, M., Bowles, J., Silvina, A., Webber, T. (2021). **On the Benefits and Security Risks of a User-Centric Data Sharing Platform for Healthcare Provision.** UMAP ’21 June 2021 Pages 351–356 doi: 10.1145/3450614.3464473

[19] Banton, M., Bowles, J., Silvina, A., Webber, T. (2021). **Model-based Security Assessment on the Design of a Patient-Centric Data Sharing Platform**. Datamod 2021.

[20] Banton, M., Bowles, J., Silvina, A., Webber, T. (2021). **Design of a Trustworthy and Resilient Data Sharing Platform for Healthcare Provision.** EDCC 2021: Dependable Computing – EDCC 2021 Workshops pp 144-151. doi: 10.1007/978-3-030-86507-8_14

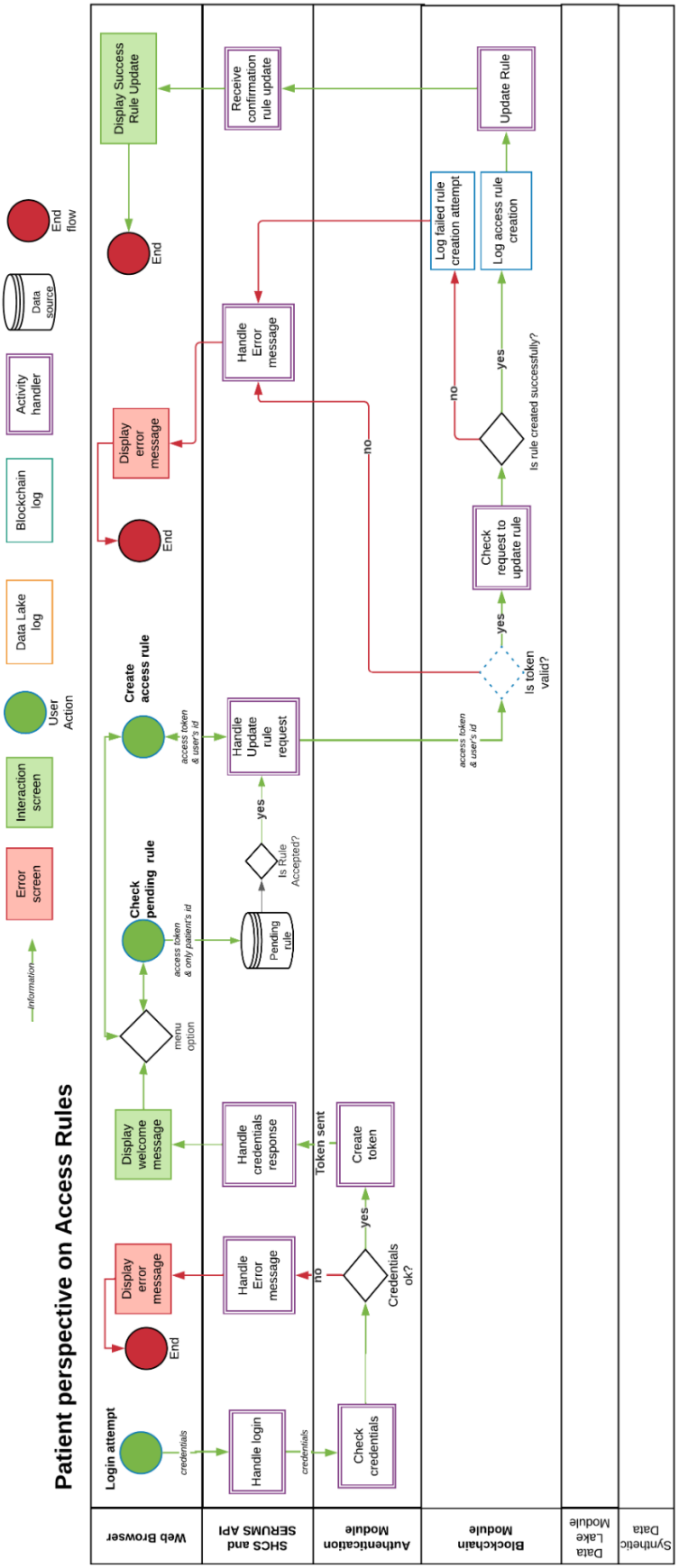
APPENDIX I - Serums Platform Information Flow Viewpoint

A. SPHR Retrieval Information Flow



Note: Patient and doctor roles for same person, it is coded with different IDs in the system, so the functions available follow the current user type logged in the system.

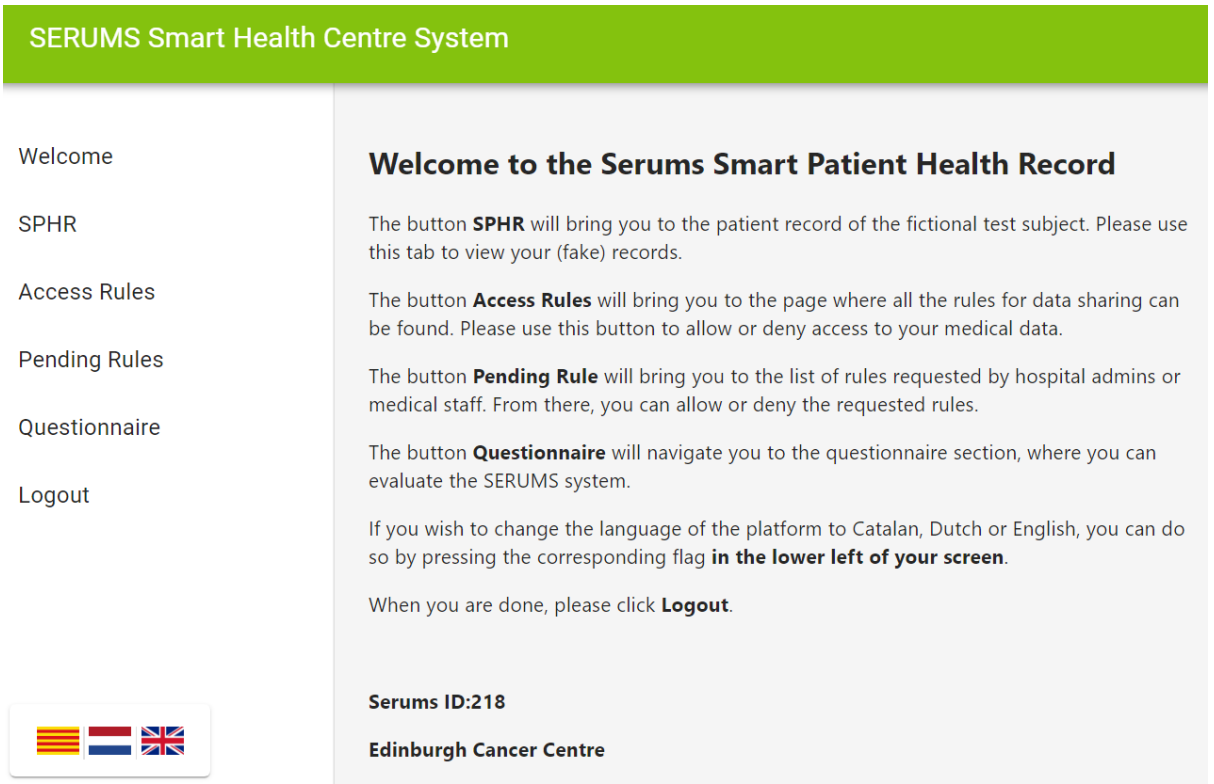
B. Access Rules Creation/Update Information Flow



APPENDIX II - Serums SHCS Web User Interface (WUI)

A. WUI - Welcome page and language translations (patients and professionals)

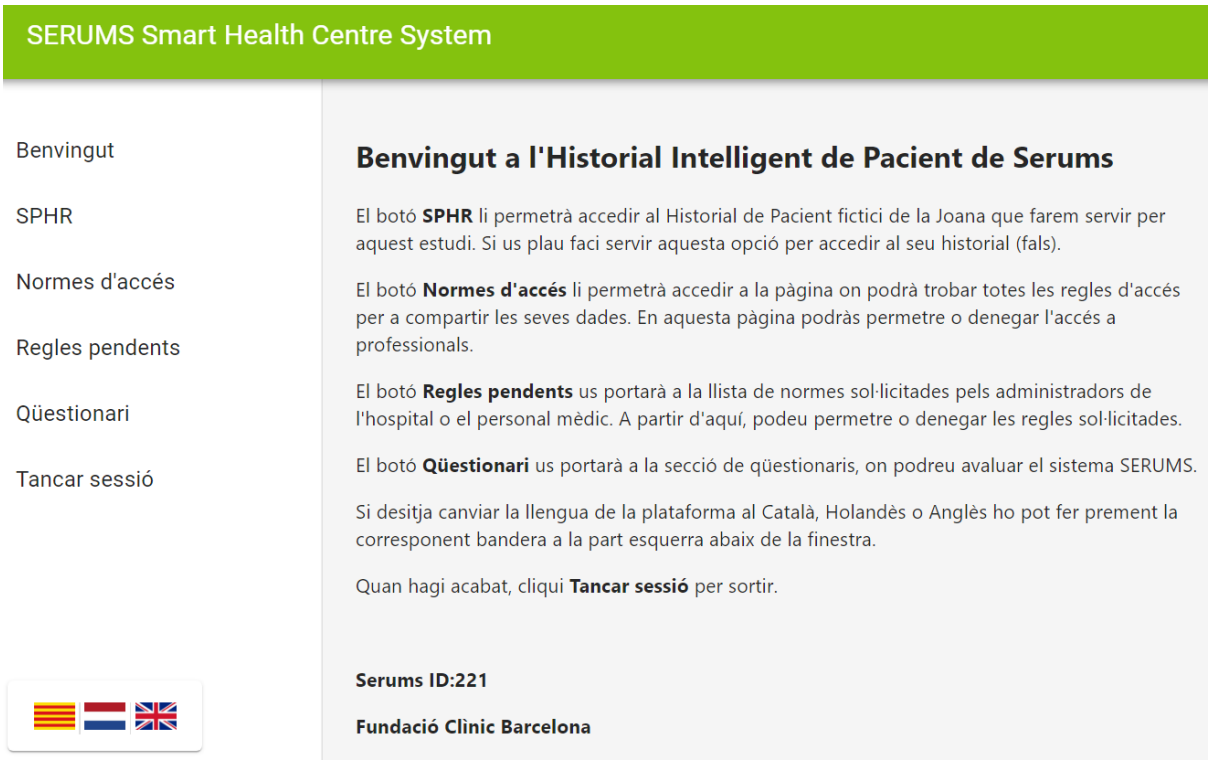
English version (patient)



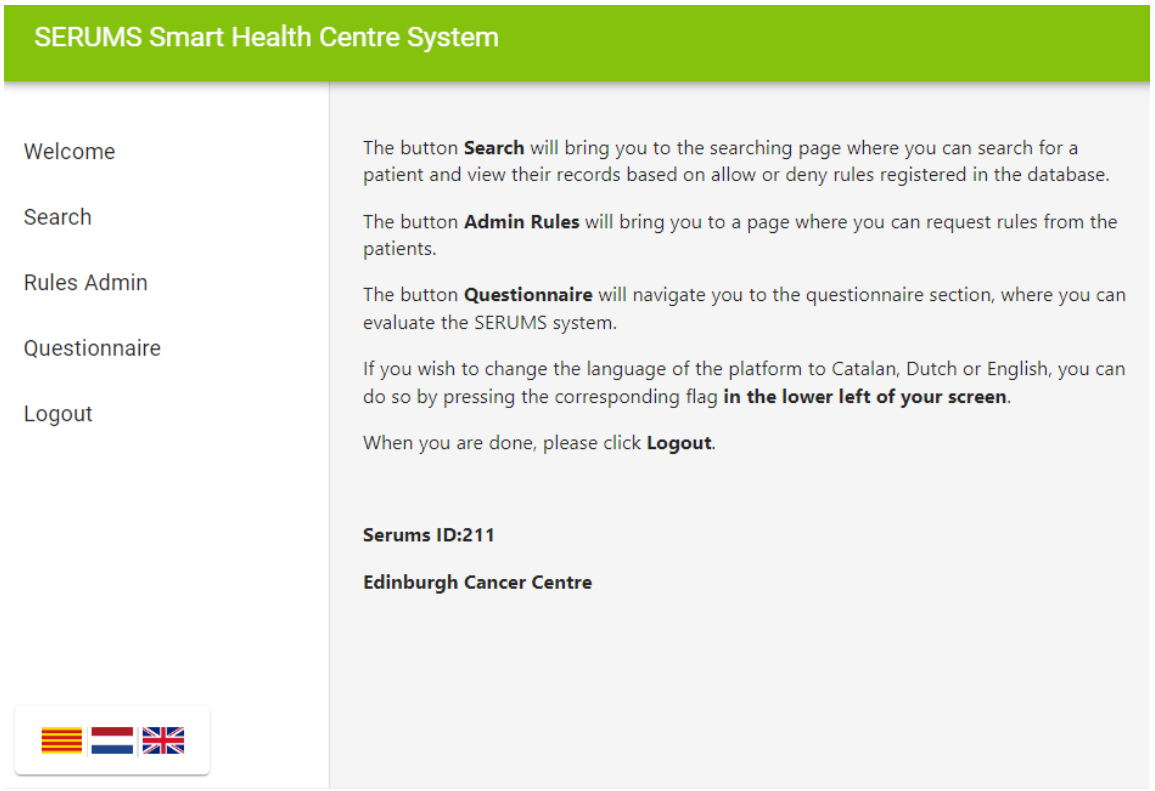
**Note: In all versions (translations), this page contains the left menu with 6 options, the bottom menu with available translations and the Welcome text containing detail on the features. This page also informs the User's Serums Id and the location informed at the Serums registration process (Authentication system - WP5).*

Dutch version (patient)

Catalan version (patient)



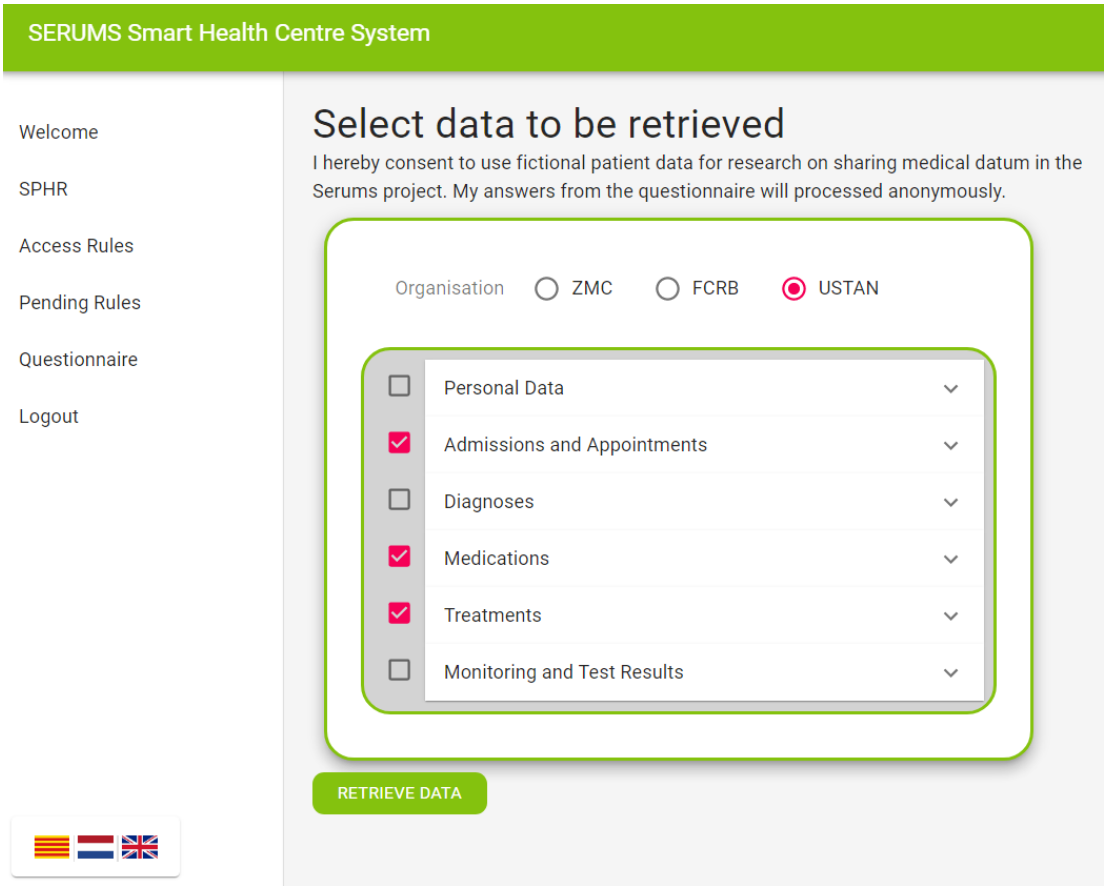
English version (professional)



**Note: this page also has translations for professionals in Dutch and Catalan.*

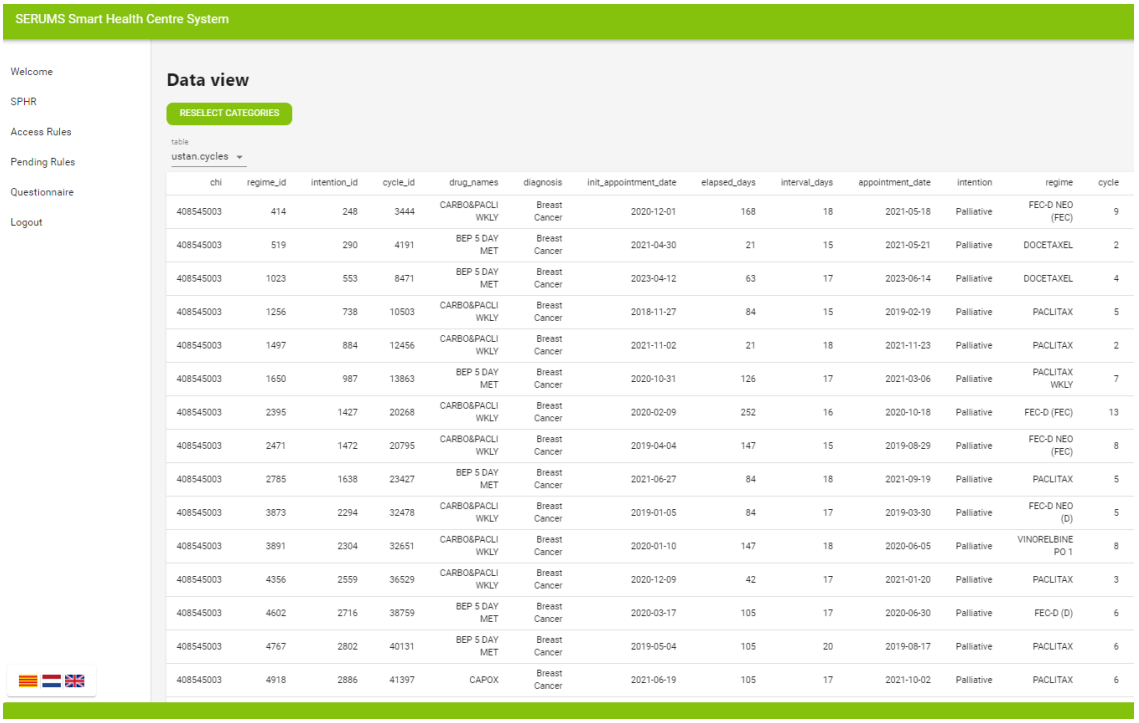
B. WUI - SPHR button feature pages (patient user)

Categories selection for SPHR Data View



**Note: this page shows how the user selects categories found in the SPHR to access, for example, medical data from Admissions and Appointments, Medications and Treatments data within USTAN organisation database.*

SPHR Data View page

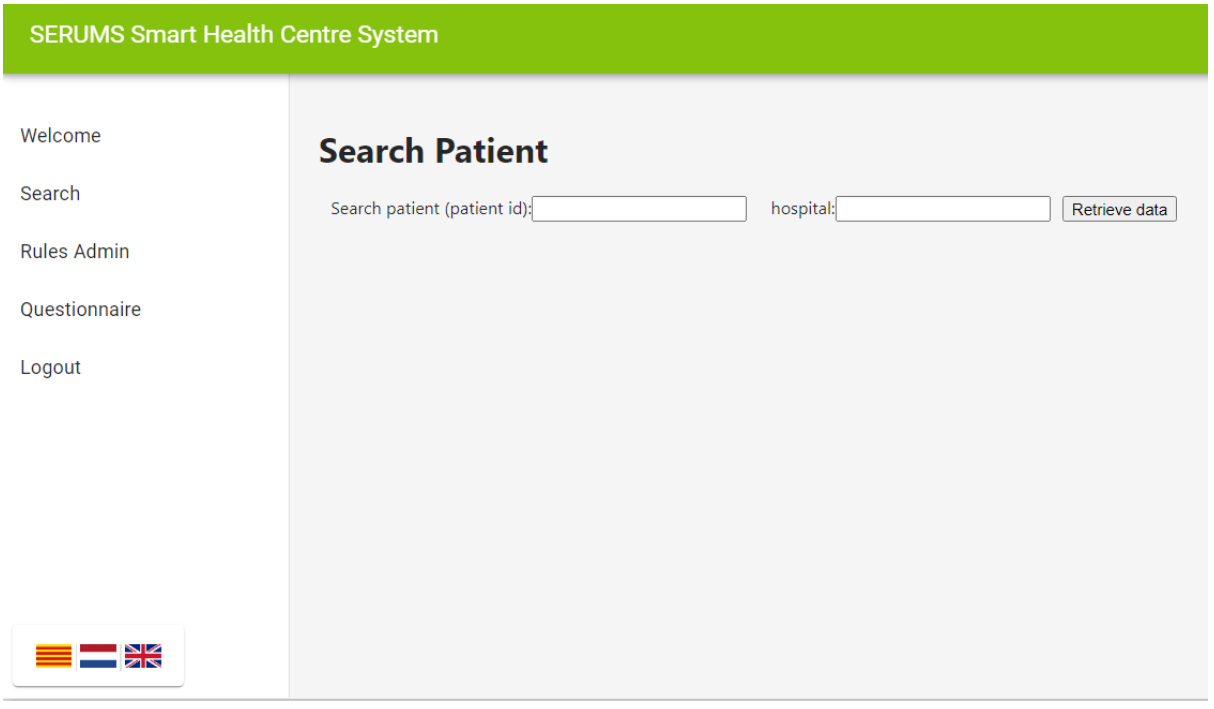


chi	regime_id	intention_id	cycle_id	drug_names	diagnosis	init_appointment_date	elapsed_days	interval_days	appointment_date	intention	regime	cycle
408545003	414	248	3444	CARBO&PACLI WKLY	Breast Cancer	2020-12-01	168	18	2021-05-18	Palliative	FEC-D NEO (FEC)	9
408545003	519	290	4191	BEP 5 DAY MET	Breast Cancer	2021-04-30	21	15	2021-05-21	Palliative	DOCETAXEL	2
408545003	1023	553	8471	BEP 5 DAY MET	Breast Cancer	2023-04-12	63	17	2023-06-14	Palliative	DOCETAXEL	4
408545003	1256	738	10503	CARBO&PACLI WKLY	Breast Cancer	2018-11-27	84	15	2019-02-19	Palliative	PACLITAX	5
408545003	1497	884	12456	CARBO&PACLI WKLY	Breast Cancer	2021-11-02	21	18	2021-11-23	Palliative	PACLITAX	2
408545003	1650	987	13863	BEP 5 DAY MET	Breast Cancer	2020-10-31	126	17	2021-03-06	Palliative	PACLITAX WKLY	7
408545003	2395	1427	20268	CARBO&PACLI WKLY	Breast Cancer	2020-02-09	252	16	2020-10-18	Palliative	FEC-D (FEC)	13
408545003	2471	1472	20795	CARBO&PACLI WKLY	Breast Cancer	2019-04-04	147	15	2019-08-29	Palliative	FEC-D NEO (FEC)	8
408545003	2785	1638	23427	BEP 5 DAY MET	Breast Cancer	2021-06-27	84	18	2021-09-19	Palliative	PACLITAX	5
408545003	3873	2294	32478	CARBO&PACLI WKLY	Breast Cancer	2019-01-05	84	17	2019-03-30	Palliative	FEC-D NEO (D)	5
408545003	3891	2304	32651	CARBO&PACLI WKLY	Breast Cancer	2020-01-10	147	18	2020-06-05	Palliative	VINORELBINE PO 1	8
408545003	4356	2559	36529	CARBO&PACLI WKLY	Breast Cancer	2020-12-09	42	17	2021-01-20	Palliative	PACLITAX	3
408545003	4602	2716	38759	BEP 5 DAY MET	Breast Cancer	2020-03-17	105	17	2020-06-30	Palliative	FEC-D (D)	6
408545003	4767	2802	40131	BEP 5 DAY MET	Breast Cancer	2019-05-04	105	20	2019-08-17	Palliative	PACLITAX	6
408545003	4918	2886	41397	CAPOX	Breast Cancer	2021-06-19	105	17	2021-10-02	Palliative	PACLITAX	6

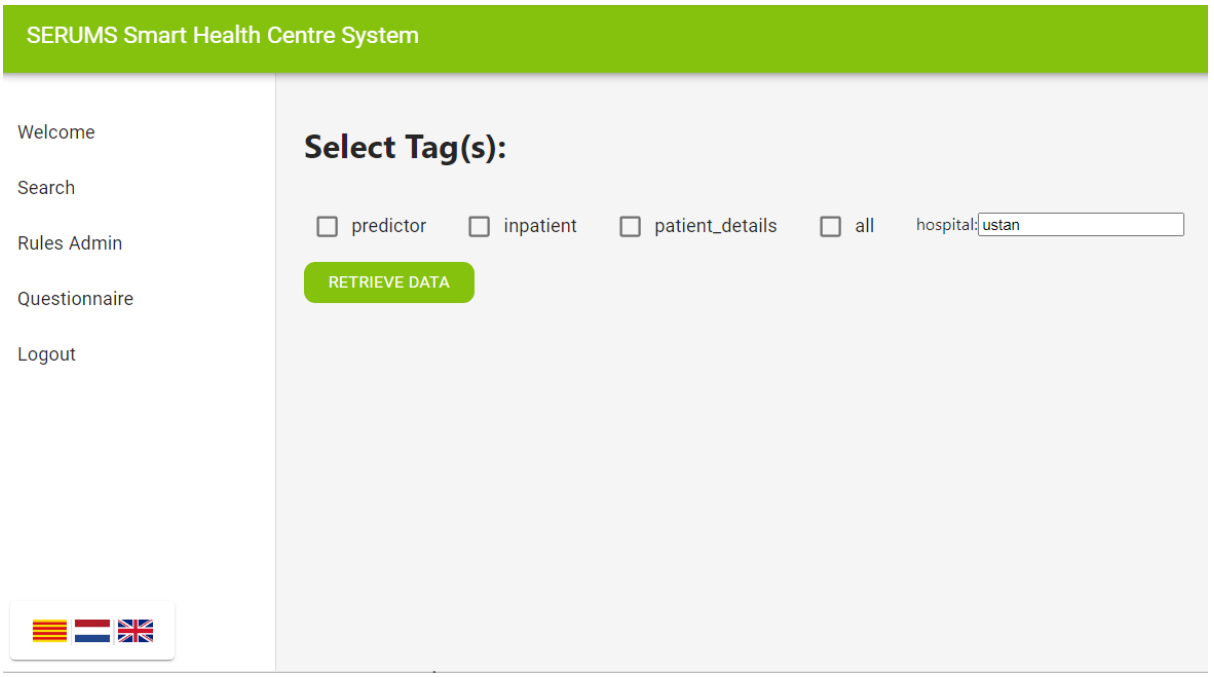
**Note: this page shows how the user can visualise the SPHR after selecting categories (i.e., the data to be retrieved)*

C. WUI - SPHR feature pages (healthcare professional user)

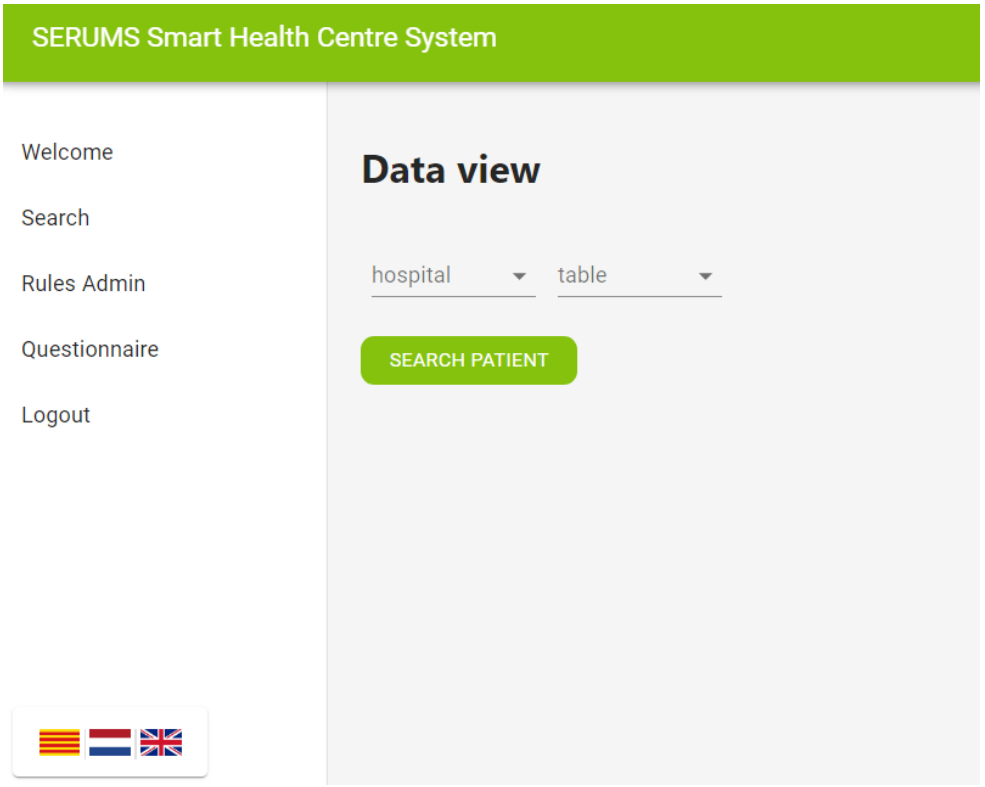
Search Patient feature



Select Patient Data to Retrieve

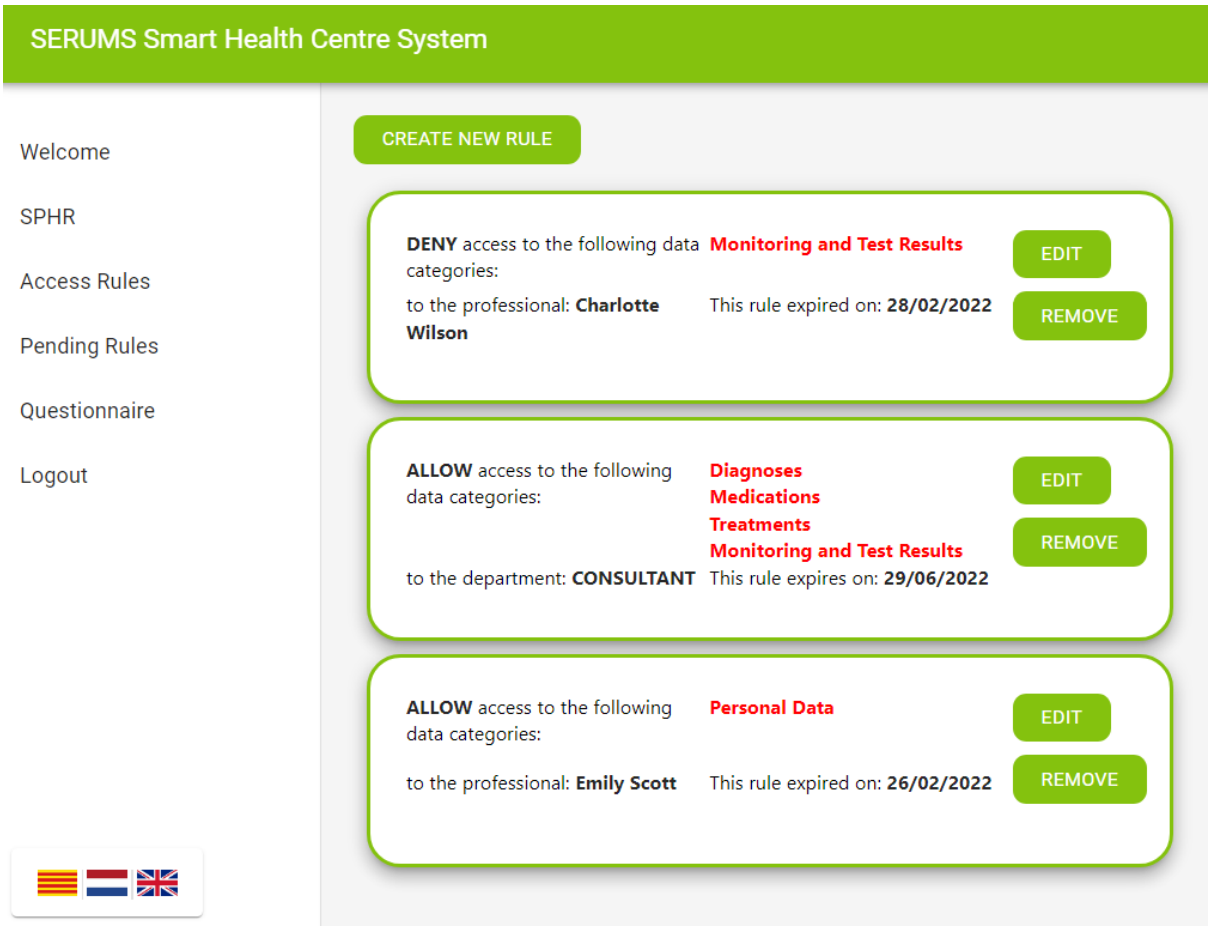


Patient Data view



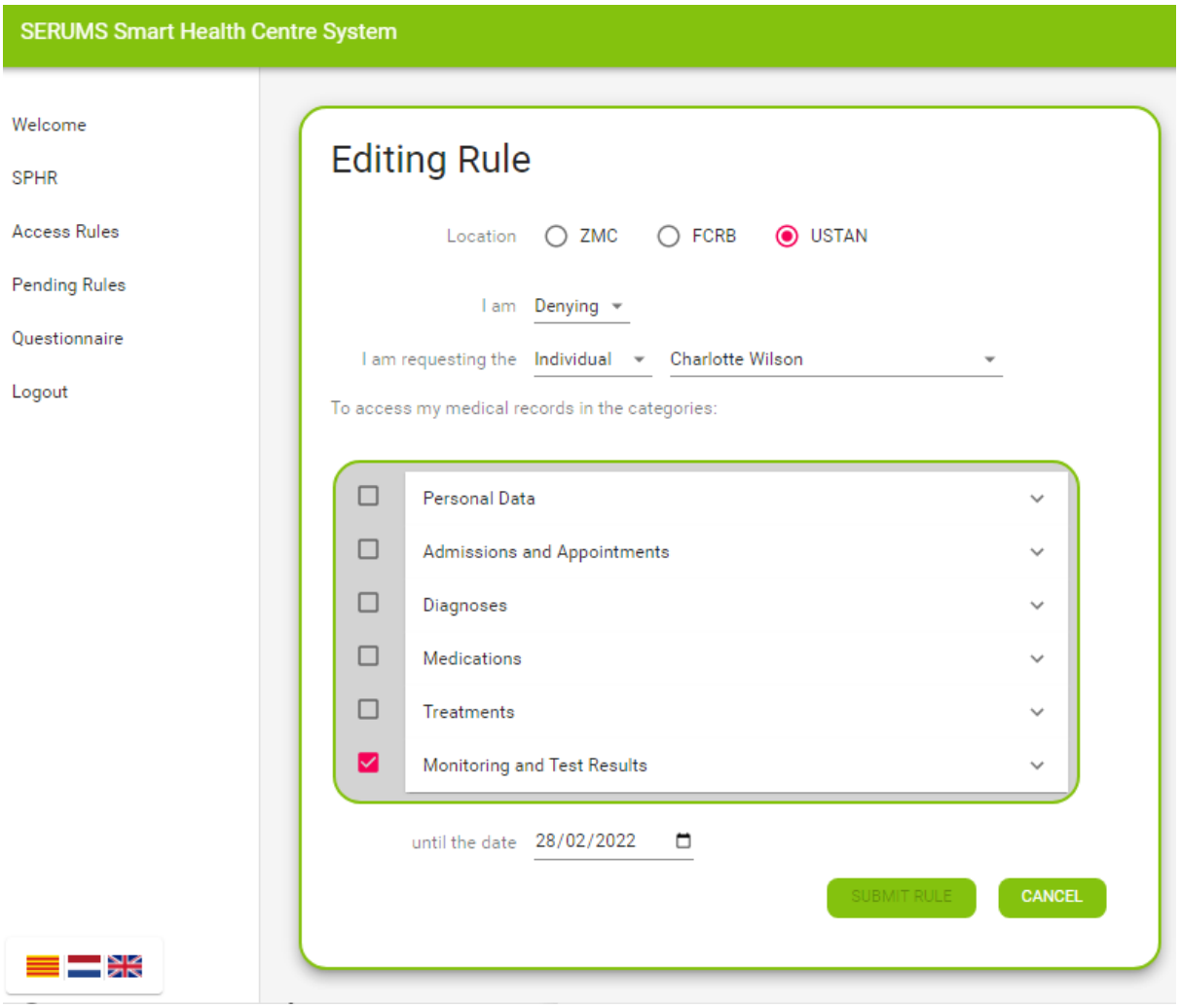
D. WUI - Rules creation pages (patient user)

Access Rules feature



**Note: this page shows the list of access rules for a patient and the available features to Edit and Remove them with immediate effect in the system. The rules information is also shown for the user, so they can quickly view the ALLOW/DENY in place for professionals as well as the data categories (in red) and the rule expiration date. The examples illustrated in this screenshot represent the professional user 'Charlotte Wilson' has been denied access to categories 'Monitoring and Test Results', whereas the whole department 'Consultant' has been granted access to 'Diagnosis', 'Medications', 'Treatments' and 'Monitoring and Test Results'. Finally, professional user 'Emily Scott' has been granted access to 'Personal Data'. If 'Charlotte Wilson' was part of the 'Consultant' department, she would have access to 'Diagnosis', 'Medications' and 'Treatments', with the deny rule active on 'Monitoring and Test Results' taking priority.*

Edit Rule button feature

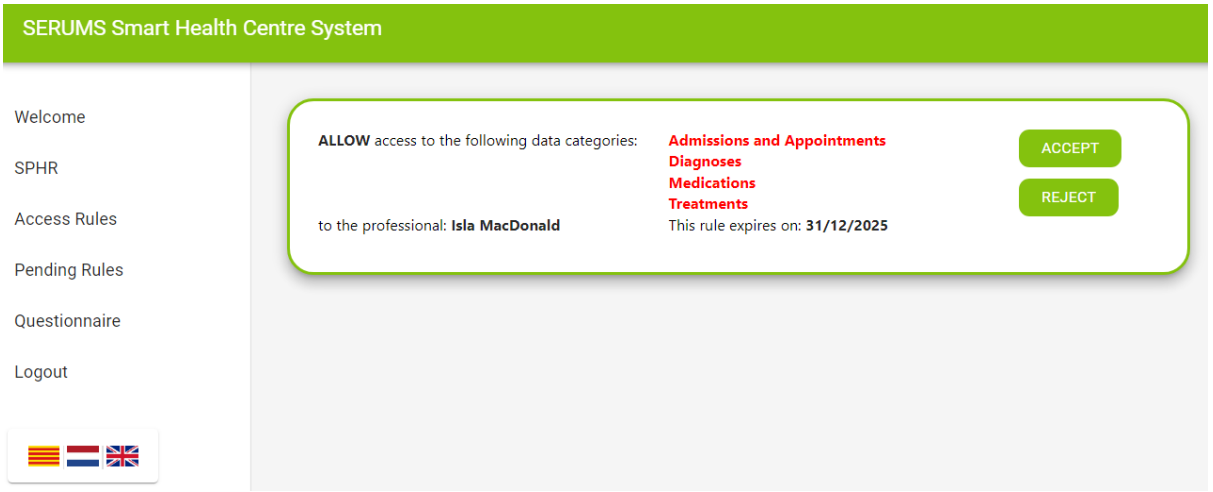


**Note: this page shows the Editing Rule form, where patients can update/create an access rule to allow or deny access to professionals or departments indicated in the "I am requesting the…" fields. The particular name of the grantee (e.g. 'Charlotte Wilson') should be typed by the user (at least 4 characters to trigger a dropbox list of names) in the respective field.*

E. WUI - Pending Rules pages (patient and professional users)

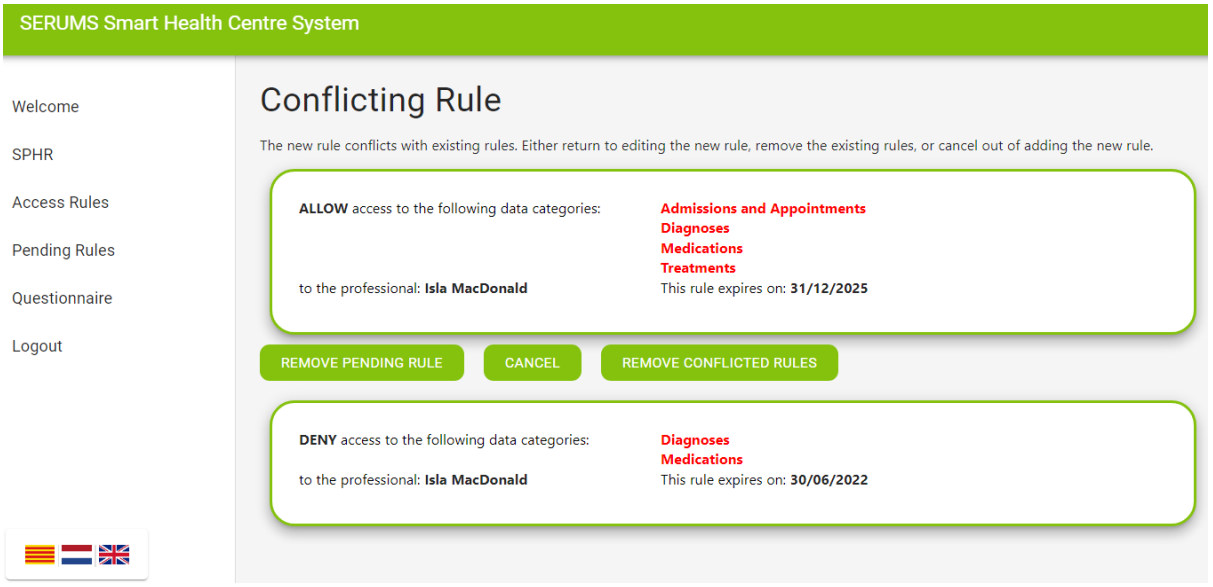
Pending Rules feature (patient)

a. Pending rules list feature



**Note: this page shows the patient can 'Accept' or "Reject' access rules requested by professionals. These requests appear in the 'Pending Rules' menu option. If none are pending, a message to the user is shown in the screen 'No pending rules'.*

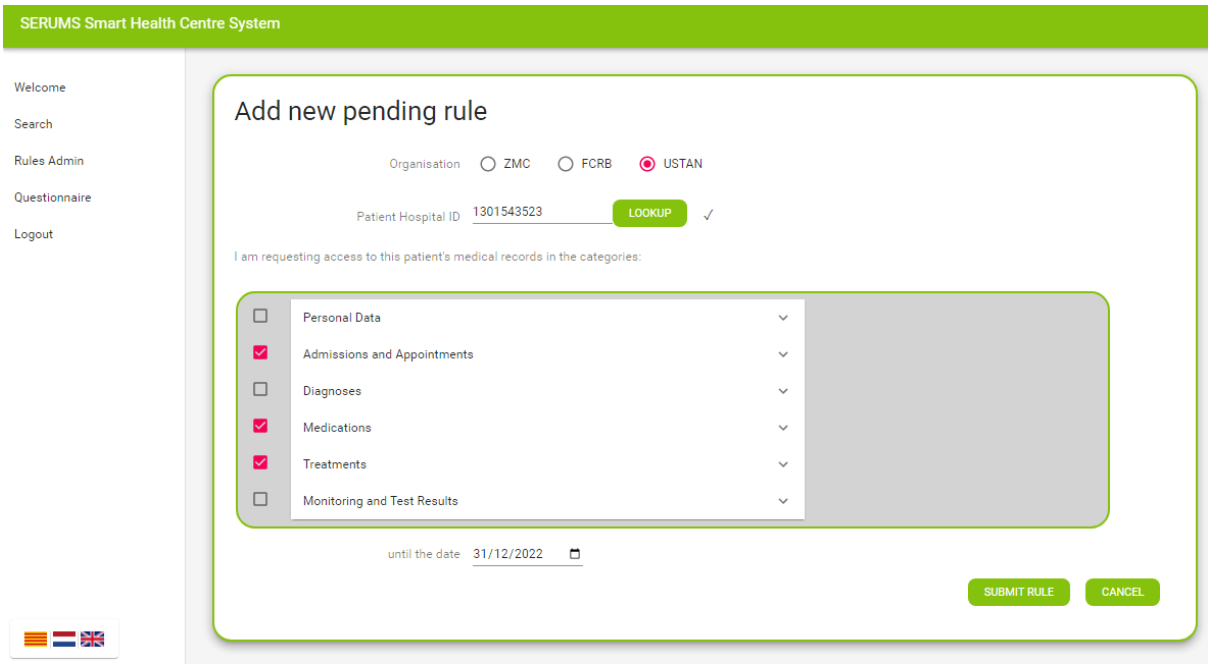
b. Conflicting Rules (patient) - On Access Rules (creation) and on Pending Rules (accept)



**Note: this page shows Conflicting Rule checker once a rule is submitted and current rules present conflicting information. The user (patient) in this case can remove the 'Pending rule' or 'Remove the conflicted rule(s)', solving the conflict in this user-friendly way.*

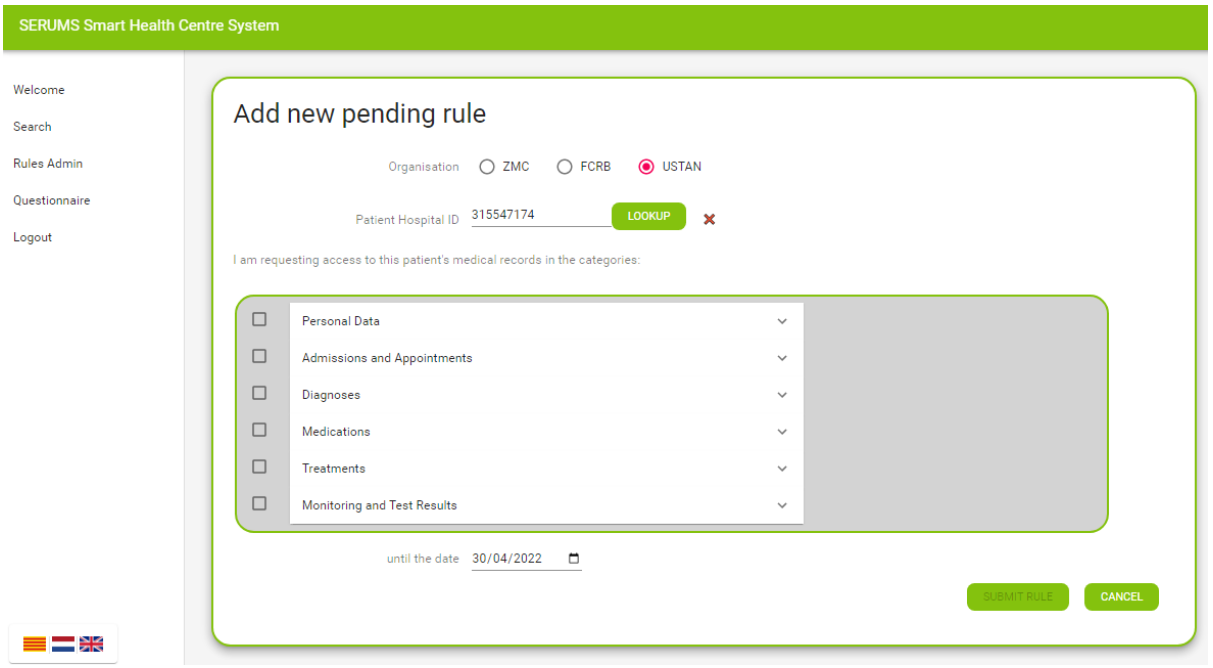
Rules Admin feature (professional)

a. Adding new Pending Rule - Patient hospital ID Lookup (ID correct or existent)

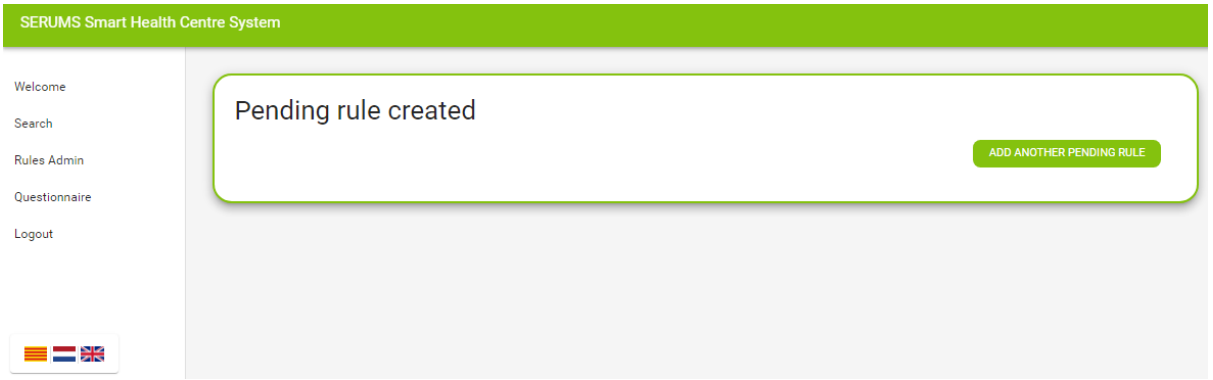


**Note: this page shows the professional page to request an access rule to a particular patient. In this case the professional needs to type the Patient Hospital ID to be able to create a request. The Lookup button searches in the database for this ID and informs the user if a valid ID is entered. Below is the same page, but showing the Lookup button informing that the entered ID is not valid.*

b. Adding new Pending Rule - Patient hospital ID Lookup (ID incorrect or inexistent)

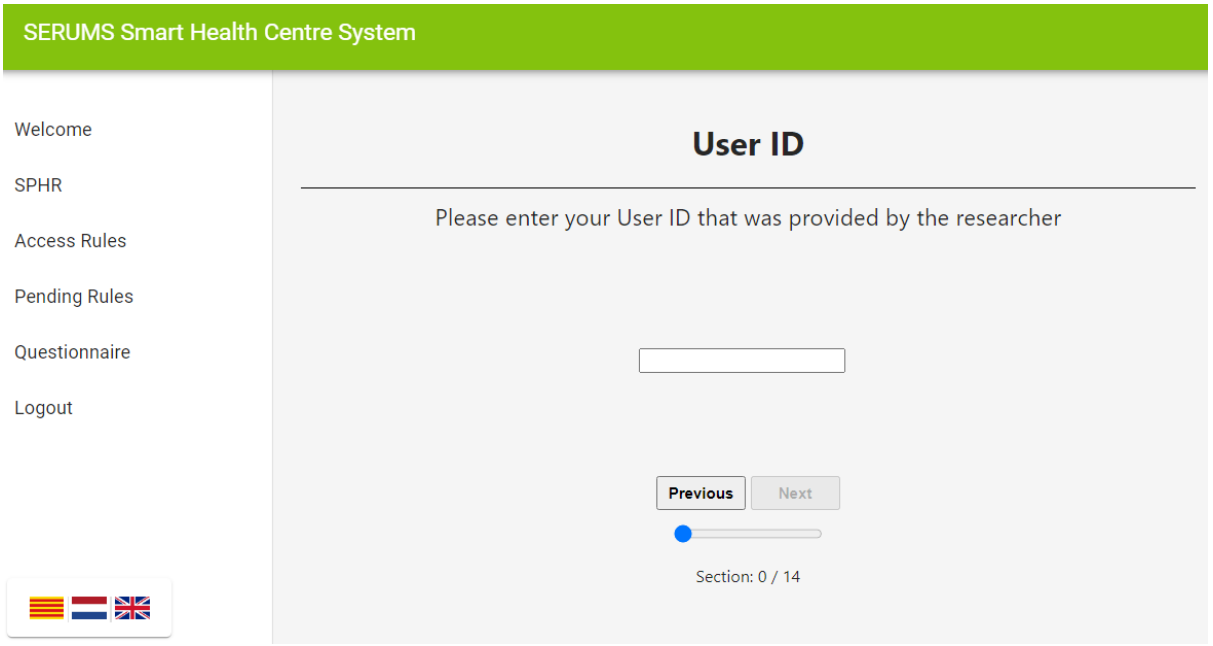


c. Adding new Pending Rule - confirmation message (professional)

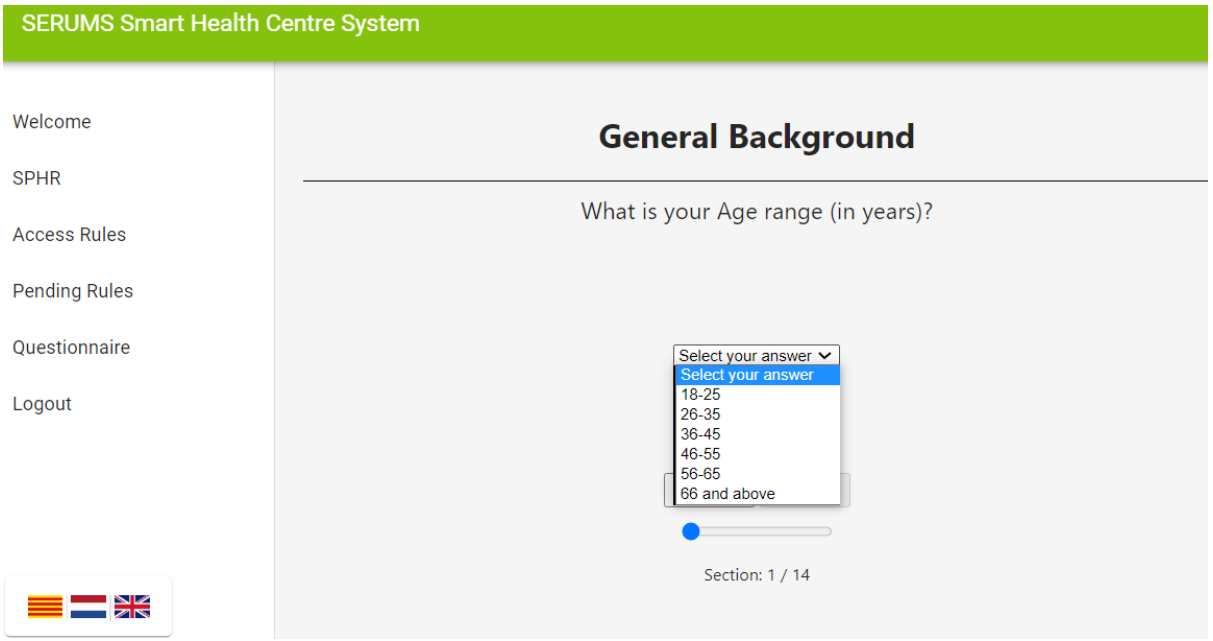


F. WUI - Questionnaire pages (main form functions included)

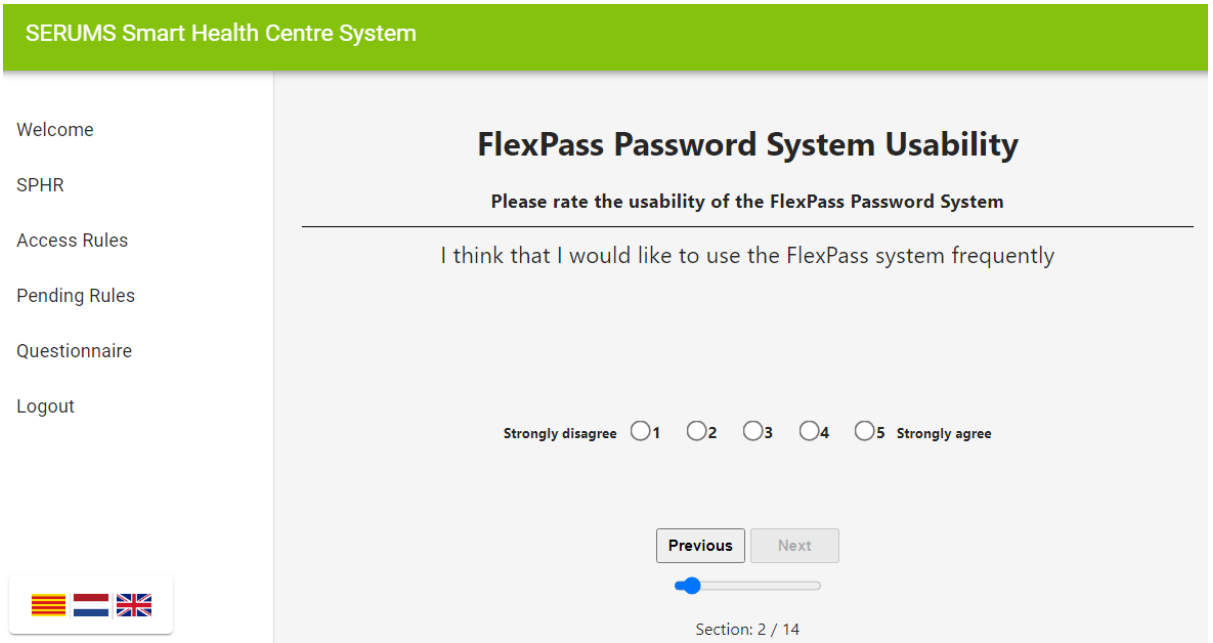
Questionnaire - first page, general layout, Text field



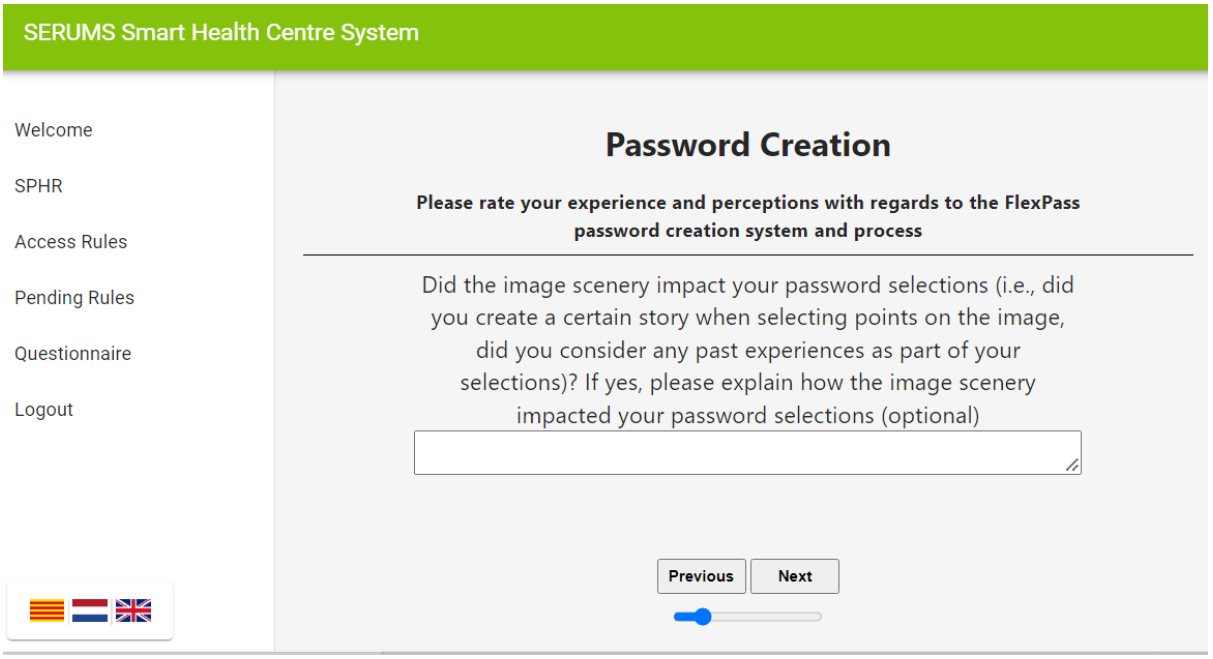
Questionnaire - Selecting answers from drop down list



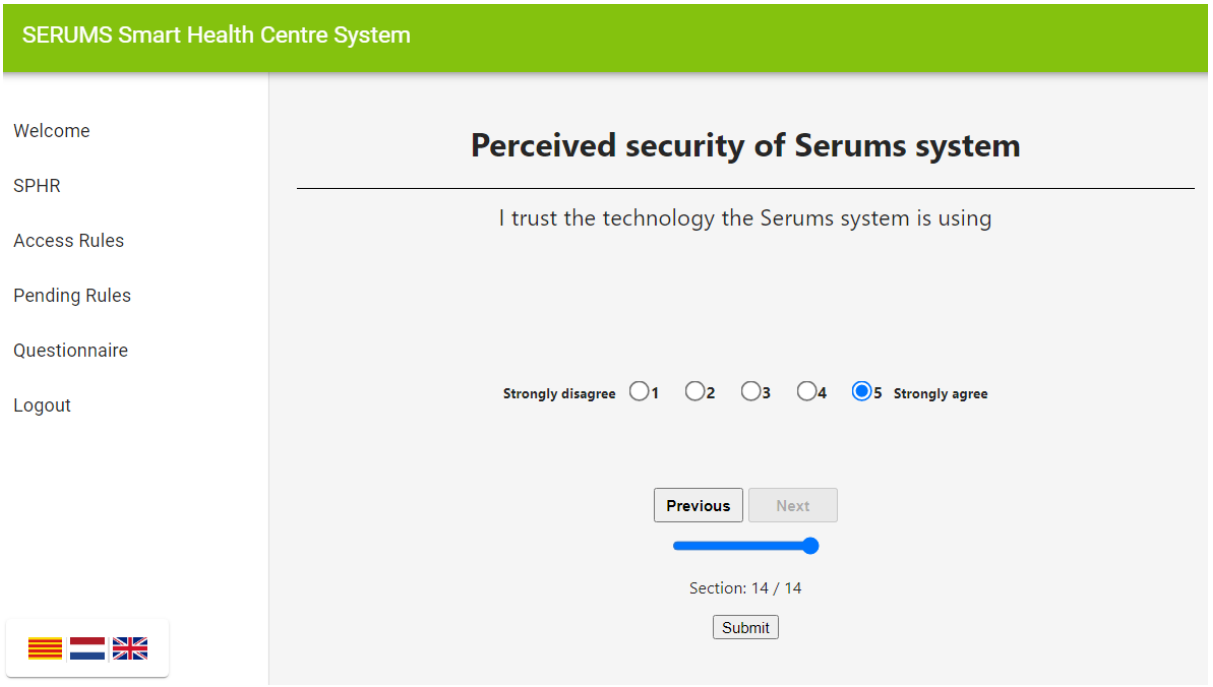
Questionnaire - Selecting answers from checkbox



Questionnaire - Text area



Questionnaire - Submit button - last page



APPENDIX III - Software libraries included

Serums SHCS Libraries versions

```
  @date-io/core": "^1.3.6",
  @date-io/date-fns": "1.3.11",
  @date-io/moment": "^1.3.13",
  @material-ui/core": "^4.9.8",
  @material-ui/icons": "^4.9.1",
  @material-ui/lab": "^4.0.0-alpha.54",
  @material-ui/pickers": "3.2.10",
  @testing-library/jest-dom": "^4.2.4",
  @testing-library/react": "^9.3.2",
  @testing-library/user-event": "^7.1.2",
  "axios": "^0.19.2",
  "date-fns": "2.11.1",
  "fernet": "^0.3.1",
  "js-cookie": "^2.2.1",
  "moment": "^2.25.3",
  "node-forge": "^0.9.1",
  "pretty-ms": "^7.0.0",
  "query-string": "^6.13.5",
  "react": "^16.13.0",
  "react-query": "^3.34.16",
  "react-cookie": "^4.0.3",
  "react-dom": "^16.13.0",
  "react-material-ui-form-validator": "^2.0.10",
  "react-router-dom": "^5.2.0",
  "react-scripts": "3.4.0",
  "use-global-hook": "^0.2.1",
  "web-vitals": "^0.2.4"
```

Serums API Libraries versions

```
Django>=2.0,<3.0
psycopg2>=2.7,<3.0
djangorestframework>=3.11.0
django-rest-swagger
django-cors-headers
django-rest-enumfield
```