Interreg
ADRION
ADRIATIC-IONIAN
EUROPEAN UNION
European Regional Development Fund - Instrument for Pre-Accession II Fund

SI4CARE

Social Innovation for integrated health CARE of ageing population in ADRION

DT2.2.1 – SI-DSS set up and implementation

T2: Social Innovation in Healthcare services: tools and pilots for best cases in action

Activity T2.2: Social Innovation Decision Support System SI-DSS development

Document	Best Cases
Status	Final
Version	1
Issue Date	18. 2. 2021

Project Partner	Contact Person
University of Ljubljana	*Darja Šemrov (darja.semrov@fgg.uni-lj.si)*
Jozef Stefan Institute	*Bojan Blažica (bojan.blazica@ijs.si)*
Miglierina Municipality	*Pietro Hiram Guzzi (hguzzi@gmail.com)*
University of Split School of Medicine	*Katarina Vukojević (katarina.vukojevic@mefst.hr*
Teaching Institute for Public Health Split-Dalmatia County	*Željka Karin (ravnateljica@nzjz-split.hr)*
Health Insurance and Reinsurance Institute of Federation of BiH	*Vlatka Martinović (martinovic.v@zzofbih.ba)*
National and Kapodistrian University of Athens	*Sokratis G. Papageorgiou (sokpapa@med.uoa.gr)*
Public Health Institution "Health Center" Tivat	*Jovanka Vučetić (dztivat@gmail.com)*
Special hospital for treatment and rehabilitation Merkur	*Srdjan Kožetinac (srdjan@vrnjcispa.rs)*
Regional development fund of Central Macedonia	*Chrysanthi Kiskini (c.kiskini@rdfcm.gr)*

LINK TO THE SI-DSS: https://www.si4care.eu/

TABLE OF CONTENTS

1. Brief introduction

The SI4CARE project will contribute to the creation of a transnational effective ecosystem for the Social Innovation application in integrated healthcare services for the ageing population in ADRION through a joint collaboration network and a shared strategy translated into regional and national action plans, implemented and monitored within pilots in telemedicine and accessibility to healthcare facilities, once innovative approaches have been tested and backboned by an ICT Decision Support System.

The Municipality Of Miglierina (MOM), will provide technical specification and will coordinate the development and integration of Social Innovation Decision Support System (SI-DSS), which will collect data and information from PPs' pilots and other possible sources (existing ICT solutions applied by PPs, ASPs or by external stakeholders).

A decision support system (DSS) is an information system that supports business or organizational decision-making activities.

DSSs serve the management, operations, and planning levels of an organization (usually mid and higher management) and help people make decisions about problems that may be rapidly changing and not easily specified in advance — i.e., unstructured and semi-structured decision problems. Decision support systems can be either fully computerized or human-powered, or a combination of both.

The SI4CARE partnership has decided to apply the International Classification of Functioning, Disability and Health (ICF) as model, first of all, to represent knowledge in the DSS.

The ICF received approval from all 191 World Health Organization (WHO) member states on May 22, 2001.

Why use ICF?

The ICF biopsychosocial model of functioning and disability integrates the medical and social models, functioning and disability are umbrella terms encompassing body functions, body structures, activities and participation and they are the result of the interaction between the person's health condition and both personal and environmental factors.

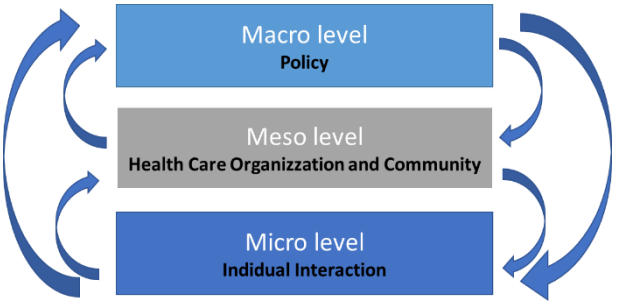

Figure 1 - Use of the ICF at different levels

Moreover, functioning and disability are made up of multiple domains. These domains are not dichotomous, but rather continuous.

The ICF is a multidimensional model, allowing the description of multiple continuous domains of functioning. Diagnosis of disease as well as information about functioning are required to comprehensively understand a person's health care, social welfare and personal needs.

The ICF provides a framework for comprehensively describing a person's individual functioning profile that in turn helps to better understand the person's specific needs.

2. General features

The aim is to create a scoring system that supports stakeholders (users) in choosing the best solution (e.g. Best Practices, Policies, Pilots) for their own context or compare different approaches, evaluating the role of different factors (through ICF codes) alone and together.

The SI-DSS in addition to returning a global coefficient, identifies a map in which all the critical areas are represented and therefore it is possible to carry out an assessment of those areas that present major or minor criticalities and therefore support users in choosing the best solution in relation to their context.

SI4CARE-DSS is not designed to support the single decision (e.g. an alarm launched by a monitoring device or the monitoring of parameters that lead to an alert or a quiet situation).

Starting from needs, expressed in the Status Quo, PPs have some challenges that they would like to face, PPs are inspired by Best Practices and they start with actions on the territory that PPs monitor, these actions involve both political, health, organizational systems, "patients", doctors, etc.

All this information (represented by the ICF codes) is not entered individually in the DSS, but through the ICF questionnaires.

In a second phase the DSS evaluates the questionnaire and calculates those indicators present in the final report.

In the final report, for each Wish List, a comparison is produced between the initial situation and after implementing the Pilots, thus allowing assessing whether the pilot worked in the area, and to what extent.

Thanks to Artificial Intelligence (AI), it is possible to overcome the approach based on binary evaluation scales (true/false), notoriously static, by introducing a measurement of functional, personal and context profiles that allows to design more appropriate "care settings" in function also of the existential expectations of fragile people.

3. The Social Innovation Decision Support System (SI-DSS)

The DSS is based on Soft Computing Algorithms and uses Artificial Intelligence to produce the results (indicators and evaluations) useful for decision support.

The proposed solution is to adopt a model based on algorithms that refer to Fuzzy Logic. The aim is to create a scoring system that supports stakeholders (users) in choosing the best solution (e.g. Best Practices, Policies, Pilots) for their own context or compare different approaches, evaluating the role of different factors (through ICF codes) alone and together.

The SI-DSS in addition to returning a global coefficient, identifies a map in which all the critical areas are represented and therefore it is possible to carry out an accurate assessment of those areas that present major or minor criticalities, and therefore support users in choosing the best solution in relation to their context.

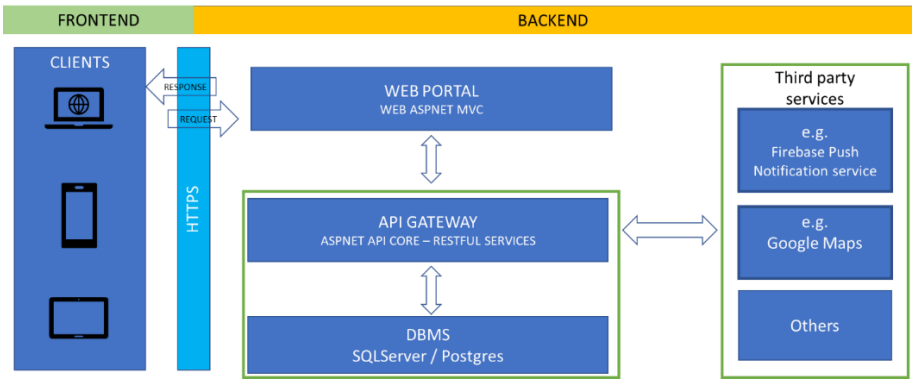

Figure 2 - System backend

Management of users and their access profiles. Through this feature, the system administrator authorizes the access of new users, defining their profile and thus granting the necessary permissions to use the system features, and thus limiting both the "what" and the "where" of operations allowed to a user.

Modification of alphanumeric data. This functionality is restricted to users based on their profile; the changes include inserting, updating and deleting data respecting the integrity constraints of the database.

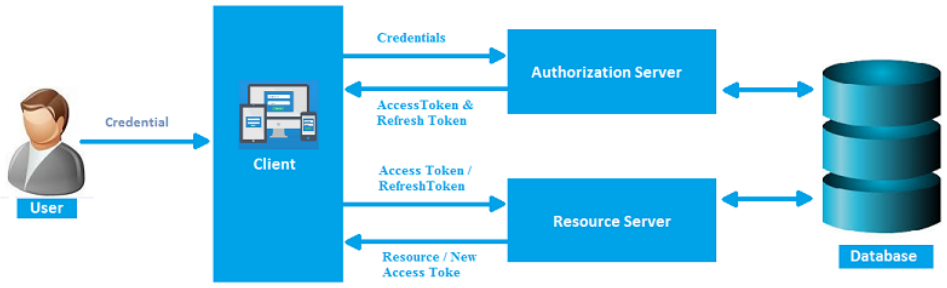

Figure 3 – System Authentication

4. SI-DSS system analysis: required features

To ensure the functionalities required by the project, the system will have to offer the following functionalities for each partner:

- Import and management of Status Quo data;
- Management of Challenges;
- Management of Best Practices;
- Management of Wish Lists;
- Management of Pilot Actions;
- Automatic generation of the Final Report, and management.

A number of other features have also been introduced, such as the ability to:

- consult the ICF at any point of the system;
- read the data of the other partners;
- share data with different partners from the same country;
- export the data.

Figure 4 shows the SI-DSS dashboard, from which the dashboard users can access the system functions listed above, and can find the Learning section, containing the manual and the system training guide.

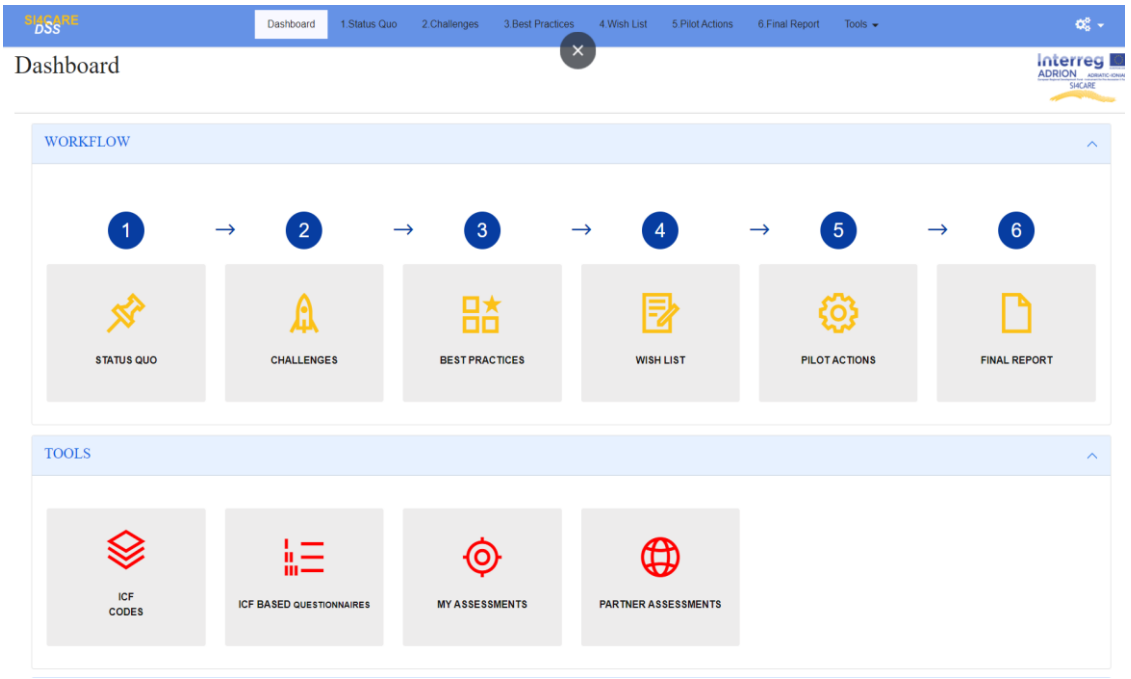

Figure 4 – SI-DSS Dashboard

4.1. Backup & Restore

The software was developed to implement the following features:

- Copy (backup) of the database.
- Restore (restore) the system starting from the previously copied data.

4.2. Development stages

The main development phases, which will be detailed in the following paragraphs, also with reference to the methods and technologies used, are as follows:

1. Detailed analysis of the requirements regarding the WEB BASED structure.
2. Application design definition of the constituent modules of the application and their detailed specifications of the system and refinement of the data model.
3. Software implementation and integration into the overall system.
4. Verification tests for the correct functioning of the entire system (unit tests, functional tests, etc.).
5. Installation and Integration of the WEB BASED structure on a dedicated server.
6. Testing verification of adherence of the product with respect to the Specifications.
7. Pre-exercise and exercise: in this phase the anomalies not found during the Testing are eliminated and evolutionary maintenance interventions are carried out.

5. Hosting

As programmed, the domains www.si4care.eu and www.si4care.cloud have been acquired in addition to the in-house servers necessary for the publication of the services developed.

6. Creating a simple data-driven CRUD service

This section explains the development of the service performing Create, Read, Update and Delete (CRUD) operations on a data source.

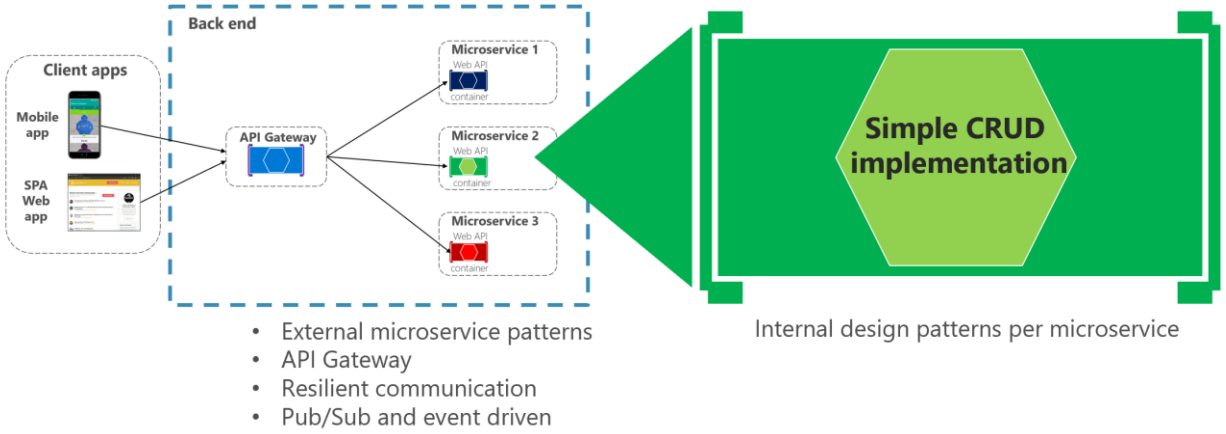

Figure 5 - Data-Driven/CRUD microservice

This type of service implements all functionality in a single ASP.NET Web API project that includes the classes for its data model, business logic, and data access code. It also stores related data in a running database.

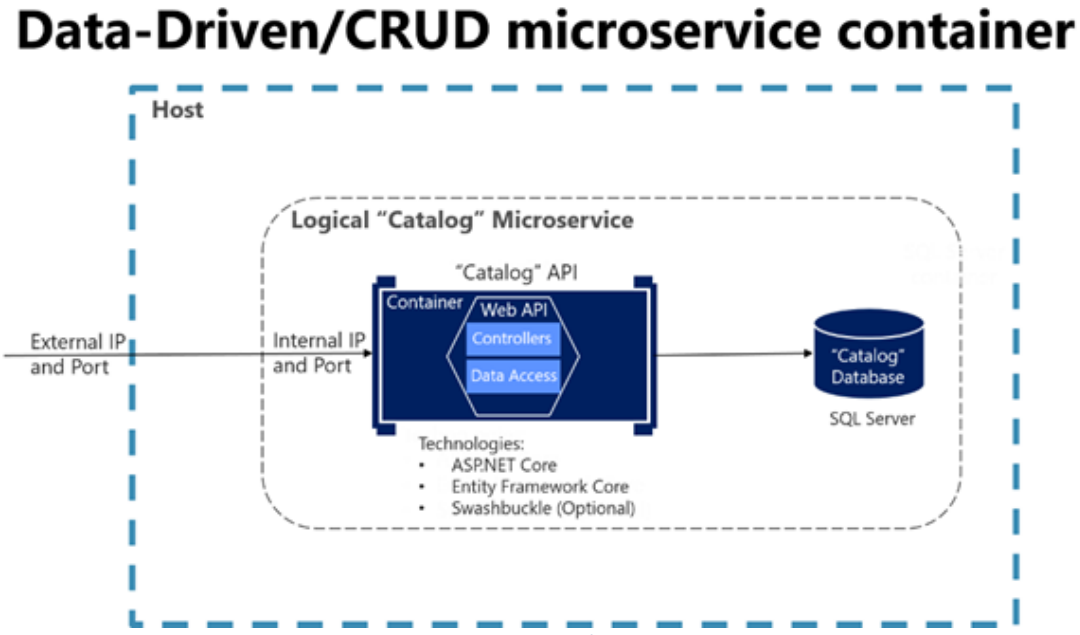

Figure 6 – Example of Data-Driven/CRUD microservice container

7. Bug tracking

A bug tracking system is a software application generally used by programmers to keep track of bug reports within software, so that these errors are kept under control, with a description of reproducibility and related details, therefore more easily solvable.

Within the development project, Bugzilla and GitLab were used as a Bug Tracking tool and is essentially composed of a database in which the descriptive elements of a bug are recorded. They can be the date on which the bug itself is reported, its severity, the incorrect behavior of the program that is affected by it, and the identity of both the person making the report and the programmer who is working on its resolution.

It is usually structured to represent the status of the bug, such as "new" if no one has yet taken over the verification or "solved" if on the contrary it has been removed and you still want to keep track of it for future consultations if the imperfection should reoccur and supports a hierarchy of users.

Users are allowed to report errors directly, thus contributing to the improvement of the product in question. The system was customized through integration with project management and customer relationship management applications.

STEPS	DESCRIPTION
Requirements analysis	Identification of the Entities, Attributes of the Entities, Relationships between Entities through interviews, possible documentary survey of the project, meetings even remotely with the project managers.
System Implementation	Implementation of DBMS PostgreSQL and MS SQL Server Development of BackOffice System with MS Visual Studio framework (C# language).
Test	Use of GIT Lab as a versioning platform and Bugzilla for bug tracking and related resolution.
Testing and pre-exercise	Identification of deficiencies in correctness, completeness and reliability of the software components under development. Verification in the execution of the software by the tester, evaluation if the behavior of the software complies with the requirements. Formal and informal testing, Performance Test, Load Testing, Stress Test.
Training and assistance	Installing and configuring the software product into the execution infrastructure that users can use. Maintenance includes changes to the software following testing, in order to correct further errors through patches (corrective maintenance), adapt it to new operating environments (adaptive maintenance or environment migration), extend its functionality (evolutionary maintenance) or convert its implementation second other structure (e.g. framework or platform migration).

7.1. Basic definitions

Malfunction (Failure)

A malfunction is the behavior of the software that deviates from the explicit or implicit requirements. In practice, it occurs when in the test environment (hardware + software) the system does not behave as expected.

Anomaly (Failure)

It is used synonymously with "malfunction" and has two definitions: (1) An accidental condition that causes a product component to fail to perform a function; (2) The manifestation of an error in the software.

Defect

It is a sequence of instructions, source or executable, which generates a malfunction when executed with particular input data. In practice, a malfunction occurs only when the code containing the defect is executed, and only if the input data is such as to highlight the error.

Error (Error, Bug)

It is used as a synonym for "defect" and has two definitions: (1) The discrepancy between the result obtained from the execution of a software function and the expected result, as foreseen by the specifications; (2) An incorrect human action causing the software to malfunction (user error in using the product).

Severity

(1) Level of inconvenience caused by an anomaly occurring in the software (2) Damage caused by the occurrence of an anomaly in the software. Depending on the discomfort or damage caused, the severity can be "high", "medium" or "low". Based on the severity, it is generally used to classify an anomaly as "blocking", "serious or non-blocking", "mild or marginal ".

8. Anomalies in flow management

The definition of the method for handling anomalies is summarized in figure 7. The management follows the use of a model for the registration of defects and their resolution.

The description of the flow follows:

- **Problem opening**
  The person who encounters the problem records the anomaly with the "Open" status and reports the basic information necessary to manage the problem (date and time, name, description of the defect detected, type of activity performed, seriousness, test phase, case of tests that detected the malfunction, other useful information such as, for example, available documentation, etc.).

- **Notification of the problem**
  The anomaly is notified to the troubleshooter (usually the development group coordinator). The notification is automatic when using an IT tool, otherwise it will have to be done manually. In any case, the time (date and time) of the notification is recorded for the purpose of detecting the resolution time of the problems.

- **Problem assignment**
  The defect resolution manager analyzes the problem to verify its validity and can proceed on two different courses of action:

  a) Acceptance of the problem

  If the problem is deemed "good", it is accepted as such and assigned to a member of the development team to solve it. This analyzes it in detail, identifies the most appropriate solution, identifies the components to be modified, makes the corrections, validates the corrections by carrying out the appropriate test cases, updates the fields relating to the resolution of the problem and updates the status of the problem as "Solved" ; the data relating to the resolution of the problem (date and time, type of problem identified, type of solution adopted, list of modified components, type of test carried out, result of the tests carried out) are stored to allow statistical analysis of the problems and to be able to intervene for process improvement.

b) Denial of the problem
If the problem is deemed "invalid", reject it (in jargon, "reject" it) and notify whoever opened the problem accordingly. If he agrees, he places the anomaly in "Closed" status, signaling that the problem has been "rejected"; otherwise he triggers a negotiation process to find an agreement. Also in this case, the information relating to the status is recorded (date and time, reason for the refusal, acceptance of the refusal).

- **Solution availability**
  The development notifies the tester of the availability of the solution by marking the problem as "Solved". Again, the date and time of the notification is recorded.

- **Validation**
  The tester re-runs the test case which detected the anomaly, verifying the correctness of the solution. Depending on the outcome of the verification, it can then proceed according to two different lines of conduct:

  a) Acceptance of the solution
  If the outcome of the test is positive, it accepts the solution and places the anomaly in the "Validated" status. The test manager will later decide to pass the status of the incident as "Closed". Also in this case the date and time of the change of status will be recorded.

  b) Rejection of the solution
  If the outcome of the tests is negative, the problem is referred to the definitive resolution group. In this case the anomaly, from the "Resolved" status in which the programmer had placed it returns to the "Assigned" status, as it was before its resolution. Also in this case, the date and time of the change of status are recorded with a comment on the reason for the refusal. Then we return to point 3 of the flow, option a). Indeed, option b) will be unlikely to occur.

The "closed" state is therefore induced by two different situations: "Problem rejected (because invalid or duplicated) or "Test with positive result".

As can be seen from the description of the flow, each change of state is recorded in terms of date and time and information that allows it to be managed.

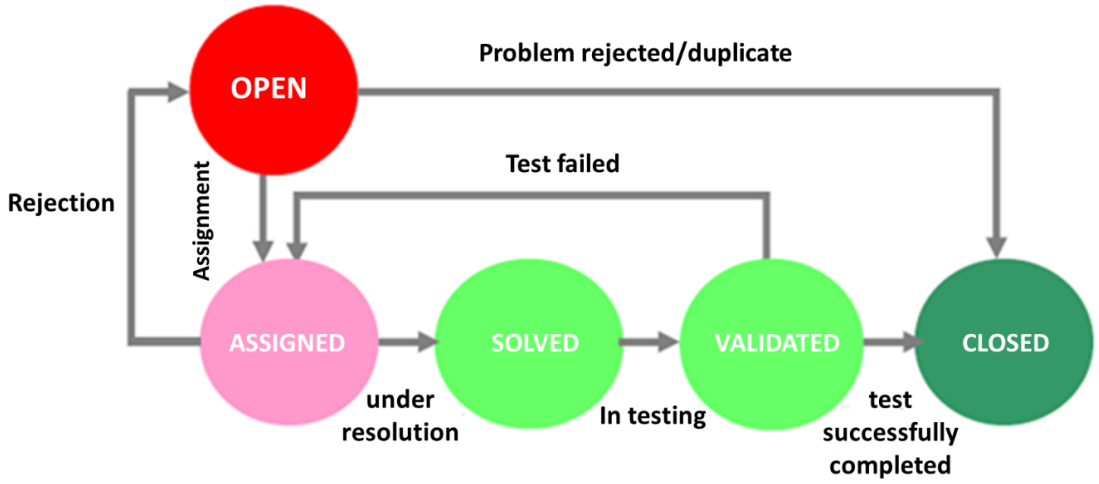

Figure 7 - Anomaly handling flow

9. IT tool for the detection of anomalies

The IT tool used to detect anomalies during the application testing phase is GitLab (de facto standard for versioning in software development) using the "Issue" function; this to allow a quick assignment of the tasks related to the resolution of the problems reported during the test phase.

The email support@si4care.eu (or any others PP-03 team email) is used to report errors or malfunctions, reports are handled by the development team as indicated above. These emails remain active for impromptu reports and for requests for clarification and/or support.

10. Detailed system features

See Annex 1 for extended functionalities of the SI-DSS. The document describes each feature of the system.

11. Case study

Annex 2 shows a simulation of use of the Si-DSS applied in a real context.