FP7-ICT-611140 CARRE

Project co-funded by the European Commission
under the Information and Communication Technologies
(ICT) 7th Framework Programme

D.7.2. Aggregator testing and RDF repository populated with test data

G. Drosatos, G. Gkotsis, R. Kizlaitis, E. Liu, A. Lukoševičius, V. Marozas, N. Portokallidis, A. Sološenko, D. Stankevičius, H. Wei

April 2015

CARRE Contacts

Project Coordinator:	Eleni Kaldoudi	kaldoudi@med.duth.gr
Project Manager:	George Drosatos	gdrosato@ee.duth.gr

DUTH Democritus University of Thrace	Eleni Kaldoudi	kaldoudi@med.duth.gr
OU The Open University	John Domingue	john.domingue@open.ac.uk
BED: Bedfordshire University	Enjie Liu	Enjie.Liu@beds.ac.uk
VULSK: Vilnius University Hospital Santariskių Klinikos	Domantas Stundys	Domantas.Stundys@santa.lt
KTU Kaunas University of Technology	Arunas Lukosevicius	arunas.lukosevicius@ktu.lt
PIAP Industrial Research Institute for Automation & Measurements	Roman Szewczyk	rszewczyk@piap.pl

Disclaimer

CARRE is a Specific Targeted Research Project partially funded by the European Union, under FP7-ICT-2013-10, Theme 5.1. “Personalized health, active ageing & independent living”.

Document Control Page

Project	
Contract No.:	611140
Acronym:	CARRE
Title:	Personalized Patient Empowerment and Shared Decision Support for Cardiorenal Disease and Comorbidities
Type:	STREP
Start:	1 November 2013
End:	31 October 2016
Programme:	FP7-ICT-2013.5.1
Website:	http://www.carre-project.eu/
Deliverable	
Deliverable No.:	D.7.2
Deliverable Title:	Aggregator testing and RDF repository populated with test data
Responsible Partner:	KTU – Vaidotas Marozas
Authors:	G. Drosatos, G. Gkotsis, E. Liu, R. Kizlaitis, A. Lukoševičius, V. Marozas, N. Portokallidis, A. Sološenko, D. Stankevičius, H. Wei
Input from:	All partners
Peer Reviewers:	Jan Piwiński, Xia Zhao, Eleni Kaldoudi
Task:	T.7.2. Aggregator testing and integration
Task duration:	5 months: 1 December 2014 to 30 April 2015
Work Package:	WP7: Integration & Evaluation
Work Package Leader:	VULSK – Romualdas Kizlaitis
Due Date:	30 April 2015
Actual Delivery Date:	6 May 2015
Dissemination Level:	PU
Nature:	P & R
Files and format:	Deliverable report: 1 pdf file Software: Updated versions of the software of all aggregators (initially delivered in D.3.2, D.3.3 and D.3.4) and RDF repositories (initially delivered in D.4.1.) populated with test data. All software versions are available from project site http://www.carre-project.eu/innovation/breakthroughs
Version:	03
Status:	☐ Draft ☒ Consortium reviewed ☒ WP leader accepted ☒ Coordinator accepted ☐ EC accepted

Document Revision History

Version	Date	Modifications	Contributors
v01.0	09 March 2015	Deliverable outline	V. Marozas
v01.1	12 March 2015	Deliverable outline review	V. Marozas, A. Lukoševičius, D. Stankevičius
v01.1	15 March 2015	Content added	V. Marozas
v01.1	16 March 2015	Content added	V. Marozas, A. Sološenko
v01.2	24 March 2015	Outline modification according to E. Kaldoudi review	V. Marozas
v01.3	30 March 2015	New content added / modifications	A. Sološenko
v01.4	31 March 2015	New content added / modifications	V. Marozas
v01.5	07 April 2015	New content added / modifications in Chapter 3	V. Marozas, D. Stankevičius
v01.5	08 April 2015	New content added / modifications in Chapter 3	A. Sološenko, V. Marozas,
v01.6	09 April 2015	Outline modification	V. Marozas, D. Stankevičius
v01.6	15 April 2015	New content added / modifications in Chapter 3	A. Sološenko
v01.7	20 April 2015	New content added in Chapters 3 and 8	G. Gkotsis
v01.8	21 April 2015	New content added in Subsection 5.3	G. Drosatos
v01.8	22 April 2015	New content added in Chapter 3 / formatting	A. Sološenko, V. Marozas
v01.9	24 April 2015	New content added in Subsection 6.2 / formatting	E. Liu, H. Wei
v02.0	25 April 2015	New content added in Subsection 5.2 / formatting	R. Kizlaitis, V. Marozas
v02.1	26 April 2015	New content added in Chapters 4, 7 / formatting	G. Gkotsis, A. Sološenko
v02.2	27 April 2015	New content added in Chapters 1, 2, finalising / formatting	V. Marozas, D. Stankevičius
v02.3	28 April 2015	New content added in Subsection 6.2 / formatting	N. Portokallidis
v02.4	29 April 2015	Response to reviews	V. Marozas
v02.5	30 April 2015	Response to reviews	All authors
v03	6 May 2015	Editing to conform to deliverable format	E. Kaldoudi

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable reports on integration and testing of the CARRE aggregators and the CARRE RDF repository population with test data (which will be used for development of visual analytics and decision support tools). First, a review of software testing methodologies was accomplished and a generic testing plan was developed. Then the generic testing plan was specialised for each aggregator. The specialised testing plans in the form of numbered test scenarios and test cases are presented together with testing results in respective tables. Testing results are discussed in short summaries for each of the tested software component. A Participant Consent Form was developed to get informed consent from healthy participants of the testing. Ten volunteers connected their devices and application based sensors to CARRE system and populated the private repository with 2.2 millions of RDF triples of personal data. The new versions of data aggregators' source code were developed as the part of the performed task at this stage of the project.

About CARRE

CARRE is an EU FP7-ICT funded project with the goal to provide innovative means for the management of comorbidities (multiple co-occurring medical conditions), especially in the case of chronic cardiac and renal disease patients or persons with increased risk of such conditions.

Sources of medical and other knowledge will be semantically linked with sensor outputs to provide clinical information personalised to the individual patient, so as to be able to track the progression and interactions of comorbid conditions. Visual analytics will be employed so that patients and clinicians will be able to visualise, understand and interact with this linked knowledge and also take advantage of personalised empowerment services supported by a dedicated decision support system.

The ultimate goal is to provide the means for patients with comorbidities to take an active role in care processes, including self-care and shared decision-making, and also to support medical professionals in understanding and treating comorbidities via an integrative approach.

Terms and Definitions

The following are definitions of terms, abbreviations and acronyms used in this document.

Term	Definition
API	Application Programming Interface - set of routines, protocols, and tools for building software applications.
App(s)	Short term casually used for Application(s)
GUID	A Globally Unique Identifier is a unique reference number used as an identifier in computer software.
HTTP	Hyper Text Transfer Protocol – message formatting and transmitting protocol used by World Wide Web
RDF	Resource Description Format - framework for how to describe any data or Internet resource and its content.
REST	Representation State Transfer – a simple stateless architecture style that generally runs over HTTP used for creating scalable web services
SPARQL	Protocol and RDF Query Language – a semantic query language able to retrieve and manipulate data stored in RDF format
SQL	Structured Query Language – programming language designed for managing data in relational databases
URI	Uniform Resource Identifier – string of characters used to identify a name of a resource
URL	Uniform Resource Locator - a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it.
XML	Extensible Markup Language – a markup language that defines a set of rules for encoding documents in a format which is both human and machine readable.

1. Introduction

This document is a report on the project Task 7.2. “Aggregator testing and integration”. It is based on and uses information provided in previous project deliverables:

- D3.2. Sensors and Aggregators for Personal Sensor Data
- D3.3. Aggregators for personal medical & lifestyle data
- D3.4. Aggregators for medical scientific/educational data from on-line sources

and provides the input to the tasks T.4.1, T.7.3.

The aim of Task 7.2 is to ensure that the (meta) data aggregators developed in WP3 are integrated, tested for functionality and bugs and populate the RDF repositories developed in WP4 with test user generated data. This data will be used subsequently for development of visual analytics and decision support services.

The report is organised in seven sections. The second section presents a review of software testing methodologies, which guides the development of a generic testing plan appropriate for this stage of the project. The third section presents the results of software testing, starting by the component “Registration and Login to CARRE System”. Specific testing plan and results for Sensors Data Aggregator testing are presented in Section 4. Section 5 presents testing plan and results of Aggregators for Personal Health Records and Life Style Data. Section 6 presents testing plan and results of Aggregators for Medical Scientific and Educational data. The final section presents performance evaluation results of CARRE RESTful API. The usage documentation of this API is disclosed in the deliverable D.4.1 “Semantic Repository Design & Implementation”.

As described in several circumstances in this document, some of the tests performed and test data collected for this deliverable involves participation of and data generation by healthy volunteers. All these volunteers were project team members, fully aware and informed about the data collection and archiving specifics and the rest of the technical issues of this projects. Additionally, they have all agreed to participate in tests and test data collection freely and have signed an appropriate consent form. The Annex 1 of the deliverable presents the Participant Consent Form developed for the volunteers participating in the data collection test. All signed consent forms are retained by the Project Coordinator.

Additional software testing for the visual analytics and decision support services is planed in forthcoming tasks: Task 5.1. “Interactive visual interface” and Task 7.3. “System and Service integration”.

2. Development of generic testing plan for aggregators software

2.1. Review of software testing methodologies and techniques

Software testing is the process of analysing a software component to detect the differences between existing and required conditions (i.e. bugs) and to evaluate the features of the software component[1]. During the years, software testing developed large body of theory, testing methodologies[2][3], techniques and standards1·4. It even became as an independent discipline in software development.

The software testing standard[4] discerns two basic classes of software testing, black box testing and white box testing. Black box testing (also called functional testing) is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. White box testing (also called structural testing and glass box testing) is testing that takes into account the internal mechanism of a system or component. The classes of testing are related to colours

[1] IEEE, "ANSI/IEEE Standard 1008-1987, IEEE Standard for Software Unit Testing," no., 1986.

[2] Hong Zhu; Yufeng Zhang, "Collaborative Testing of Web Services," Services Computing, IEEE Transactions on , vol.5, no.1, pp.116,130, Jan.-March 2012

[3] M. Kaur, R. Singh “A Review of Software Testing Techniques”, International Journal of Electronic and Electrical Engineering, ISSN 0974-2174, Volume 7, Number 5 (2014), pp. 463-474

[4] IEEE Draft International Standard for Software and Systems Engineering--Software Testing--Part 4: Test Techniques," IEEE P29119-4/DIS2-Feb2014 , vol., no., pp.1,139, April 23 2014.

(black versus white) to depict the opacity of the code to the testers. In black box testing, the software tester does not have and does not need access to the source code itself. The code is considered to be a “black box” to the tester who can’t see inside the box. The tester knows only that information can be input into to the black box, and the black box will send something back out. Based on the requirements knowledge, the tester knows what to expect the black box to send out and tests to make sure the black box sends out what it is supposed to send out. Alternatively, white box testing focuses on the internal structure of the software code. The white box tester (most often the developer of the code himself) knows what the code looks like and writes test cases by executing methods with certain parameters. Black box testing is often used for validation which is supposed to answer the question are we building the right software? White box testing is often used for verification - are we building the software right?

In between of black-box and white-box software testing approaches, some discern grey-box testing. Grey-box is a technique to test the software application when limited knowledge of the internal working (data structures and algorithms) of an application is used to design tests, while executing those tests at the user or black-box level[5]. A grey-box tester is permitted to set up an isolated testing environment with activities such as seeding database. The tester can observe the state of the product being tested after performing certain actions such as executing SQL statements against the database and then executing queries to ensure that the expected changes have been reflected. Grey-box testing implements test scenarios, based on limited information. This will particularly apply to data type handling, exception handling, and so on.

Yet another software testing methodology is exploratory testing[6]. The exploratory testing can be described as simultaneous learning, test design and test execution. The main advantage of exploratory testing is that less preparation is needed, important bugs are found quickly, and at execution time, the approach tends to be more intellectually stimulating than execution of scripted tests (test cases). Another major benefit is that testers can use deductive reasoning based on the results of previous results to guide their future testing on the fly. This also accelerates bug detection when used intelligently. Exploratory testing is particularly suitable if requirements and specifications are incomplete, or if there is lack of time.

Usually software components depend on other system parts, which are not developed yet. To support software testing in such cases auxiliary code or data – scaffolding code – may be used. This is code that simulates the functions of components that do not exist yet and allow the program to execute[7]. Scaffolding code includes writing stubs and test drivers. Stubs are modules that simulate components that aren’t written yet, formally defined as a computer program statement substituting for the inner body of a software module that is or will be defined elsewhere[7]. Test drivers are defined as software modules provide test inputs, controls, and monitor execution and report test results. The third concept to support unfinished software testing is mock objects: software code and data that are able to temporarily substitute the domain code and emulate the real code.

Test planning should be performed throughout development cycle. A test plan is a document, which identifies test items, the features and functions to be tested and other activities during testing process. Important components of the test plan are the individual test scenarios and test cases. A test case is a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. A test scenario is a group of related test cases. Some type of test planning template is usually used to organise the testing plan and testing results. The test plan template adopted in our report is shown in Table 1.

We have defined the format of tables and the naming style (no) of test scenarios and cases. The following naming style of the test scenario TS_X_Y_Z is used:

- TS: Test scenario;
- X: F or N i.e. functional or non-functional testing;
- Y: software component (aggregator) name, e.g. L - lifestyle;
- Z: a sequential number, e.g. 01.

[5] M. E. Khan, F. Khan “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques”, International Journal of Advanced Computer Science and Applications, Vol. 3, No.6, 2012.

[6] Itkonen, J.; Mantyla, M.V.; Lassenius, C., "The Role of the Tester's Knowledge in Exploratory Software Testing," Software Engineering, IEEE Transactions on , vol.39, no.5, pp.707,724, May 2013

[7] Glenford J. Myers, Corey Sandler, Tom Badgett “The Art of Software Testing”, 3rd Edition, ISBN: 978-1-118-03196-4 240 pages December 2011

As for the test cases: TC - X

- TC: Test case;
- X: a sequential number, e.g. 01.

The particular case is referred to like this: TS_F_L_01.TC – 01.

Table 1. Test scenarios and test cases of functional and non-functional testing of the X data aggregators

No	Test Scenario	Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_X_Y_Z	Test scenario description	TC-01	Test case description	Pass / Fail (description of failure)

We will use a mixture of software testing methodologies in this report and during lifetime of the project. We will use mock objects for testing software components, which depend on other components not developed, yet (e.g. visual analytics and services). We will use test planning to increase test repeatability and clarity. In summary, we should keep in mind that "*Program testing can be used to show the presence of bugs, but never to show their absence!*"[8]

2.2. Generic testing plan for all aggregators

The generic testing plan is intended for all CARRE system aggregators. The main task before following the general testing plan is to define specifications, use cases and testing environment of the software application to be tested. Generic testing plan consists of functional and non-functional testing phases:

Functional testing:

1. Functionality testing – testing of all intended functions of the certain application component. Example: password recovery functionality.
2. Integration testing – verification that all software application components interact with each other as intended. Example: aggregation of certain data types from different sources.
3. API testing – testing of the API functionality. Example: get intended output on certain API call.

Non-functional testing:

1. Scalability testing – verification of the ability to scale and expand features of the application if demanded. Example: adapt to increasing workload.
2. Data validation testing – verification that certain software application features use intended datatypes and their ranges. Example: adequate minimum and maximum ranges of the certain data type, allowed datatypes, etc.
3. Compatibility testing – verification that web application is compatible with intended platforms. Example: browser (Chrome, Opera, Firefox) compatibility.
4. Usability testing – evaluation of easiness to learn and use software application. Example: User interface control.

[8] E. W. Dijkstra, "Notes on Structured Programming," Technological University Eindhoven T.H. Report 70-WSK-03, Second edition, April 1970.

5. Security testing – verification weather web application is vulnerable to attacks and weather there are no information leaks. Example: login to the web application without authorization, unauthorized data retrieval, authorization circumvention, etc.
6. Performance testing – web application and its components response under various load conditions. Example: time taken to visualize large amount of data.

3. Testing of registration and login to CARRE system

3.1. Specific testing plan and results

This section deals with the functional testing of new user registration and login to the CARRE system. Since user registration and login are common among all the aggregators, they are tested independently from the other CARRE software application components. Table 2 summarizes test scenarios and test cases of functional testing for Registration and Login to CARRE System.

Table 2. Test scenarios and test cases of functional testing for Registration and Login to CARRE System

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_F_RL_01	User registration	TC-01	User registers	Pass
		TC-02	Cancel submission of information for registering	Pass
		TC-03	Provide insufficient data	Pass
		TC-04	Register existing user	Pass
TS_F_RL_02	User login	TC-01	Normal login	Pass
		TC-02	Cancel submission of login information	Pass
		TC-03	Provide insufficient data	Pass
		TC-04	Account recovery procedure	Pass

3.2. Summary of testing results

All intended functions of the user registration and login module are functioning as planed except for the verification of the new user registration, which is not implemented yet. There are some minor bugs, which occur after logging into the CARRE system, specifically error messages:

error trapped in error: function(msg, url, line);

msg = [object Object], url = error, line =;

No security issues during logging in and recovering account were noticed during the testing.

4. Testing of sensors data aggregators

This section deals with functional and non-functional testing of the sensor data aggregator. The aim of the functional testing is to verify aggregator readiness to execute functions and user experience covered in the functional requirements and specifications. The testing includes functions such as user commands, data manipulation and fetching from different 3rd party clouds, integration etc. The aim of the non-functional testing is the verification of the sensor data aggregators' quality characteristics such as usability, performance, security, etc. The task of non-functional testing is to evaluate the readiness of a system according to the various criteria (based on the requirements), which are not covered by functional testing and demonstrate sensor data aggregators' behaviour.

4.1. Specific testing plan and results

Table 3 presents functional testing results of the sensor aggregator.

Table 3. Test scenarios and test case of functional testing of Sensor Data Aggregator

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_F_S_01	Subscription to 3rd party sensor / app data cloud	TC-01	Fitbit	Pass[9]
		TC-02	Withings	Pass[9]
		TC-03	iHealth	Pass[9]
		TC-04	Misfit	Pass[9]
		TC-05	Google Fit	Pass[9]
		TC-06	Medisana	Pass[9]
		TC-07	Moves	Pass[9]
		TC-08	Google Calendar	Pass[9]
TS_F_S_02	Unsubscription of 3rd party sensor / app data cloud	TC-01	Fitbit	Pass[10]
		TC-02	Withings	Pass[10]
		TC-03	iHealth	Pass[10]
		TC-04	Misfit	Pass[10]
		TC-05	Google Fit	Pass[10]
		TC-06	Medisana	Pass[10]
		TC-07	Moves	Pass[10]
		TC-08	Google Calendar	Pass[10]
TS_F_S_03	Integration with 3rd party sensor / App data cloud	TC-01	Fitbit	Pass[11]
		TC-02	iHealth	Pass[11]
		TC-03	Misfit	Pass[11]
		TC-04	Google Fit	Pass[11]
		TC-05	Medisana	Pass[11]
		TC-06	Google Calendar	Pass[11]
TS_F_S_03	SPARQL query	TC-01	Get all data for specific user	Pass
		TC-02	Get 100 randomly selected RDF triples for specific user	Pass

[9] Pass – the label near the sensor provider icon in the dashboard of CARRE Devices changes to "Connected". Fail – no change.

[10] Pass – the label near the sensor provider icon in the dashboard of CARRE Devices changes to "Connect". Fail – no change.

[11] Pass – the data from sensor provider is available in CARRE Private repository, Fail – not available.

		TC-03	Get all steps values for specific user	Pass
		TC-04	Get manufacturers names from which specific user gets step counts	Pass
		TC-05	Get Fitbit measured step counts for specific user	Pass
		TC-06	Get iHealth measured step counts for specific user	Pass
		TC-07	Get Misfit measured step counts for specific user	Pass
		TC-08	Get Google Fit measured step counts for specific user	Pass
		TC-09	Get Medisana measured step counts for specific user	Pass
		TC-10	Get average step count across all manufacturers for specific user	Pass
		TC-11	Get average Fitbit step count for specific user	Pass
		TC-12	Get average iHealth step count for specific user	Pass
		TC-13	Get average Misfit step count for specific user	Pass
		TC-14	Get average Google Fit step count for specific user	Pass
		TC-15	Get average Medisana step count for specific user	Pass
TS_F_S_04	CARRE Devices API	TC-01	POST /query	Pass
		TC-02	POST /deleteUser	Pass
		TC-03	GET /measurement	Pass
		TC-04	GET /measurementsList	Pass
		TC-05	GET /dailyMeasurement	Pass
		TC-06	GET /userProfile	Pass
		TC-07	GET /authenticate	Pass
		TC-08	GET /ical.ics	Pass
		TC-09	GET /icalToken.ics	Pass

Table 4 presents non-functional testing results of the sensor aggregator.

Table 4. Test scenarios and test cases of non-functional testing of sensor data aggregator

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_N_S_01	Data validation testing	TC-01	Allowed data character check	Pass
		TC-02	Data type check	Pass
		TC-03	Data consistency check	Pass

		TC-04	Data range check	Pass
		TC-05	Data limit check	Pass
TS_N_S_02	Compatibility with different internet browsers	TC-01	Desktop Opera browser	Pass
		TC-02	Desktop Chrome browser	Pass
		TC-03	Desktop Firefox browser	Pass
		TC-04	Desktop Internet Explorer	Pass
		TC-05	Android Browser	Pass
		TC-06	Android Opera	Pass
		TC-07	Android Opera Mini	Pass
		TC-08	Android Chrome	Pass
		TC-09	Android Firefox	Pass
TS_N_S_03	User interface testing	TC-01	UI content check	Pass
		TC-02	UI images check	Pass
		TC-03	UI instructions check	Pass
		TC-04	UI navigation check	Pass
		TC-05	UI usability check	Pass
		TC-06	URL manipulation	Pass
TS_N_S_04	Security testing	TC-01	URL manipulation	Pass
		TC-02	SPRQL injection	Pass
TS_N_S_05	Performance testing	TC-01	Data visualization performance	Pass

4.2. Sensor data and meta-data populated in private repository

Ten subjects (project participants) were recruited to generate sensor data and populate it to private repository (Table 5). Each participant filled in and signed the Participant Consent Form (see Annex 1). All participating subjects created private accounts in CARRE system[12] and connected their devices accounts in commercial cloud services with CARRE. All sensor devices were purchased by using CARRE budget except Apps based sensors, which were private smart phones with installed Google Fit and Moves applications for physical activity monitoring.

Data was being generated and pushed/stored in the CARRE Private repository for more than 1 month - (18/03/2015 - 22/04/2015).

[12] https://carre.kmi.open.ac.uk/devices

Table 5. Distribution of sensor devices among the test trial subjects

Subject	Weight and body composition	Physical activity,	Glucometer	Blood pressure	Electrocardiography AF
v*****m	Fitbit Aria	Fitbit One, Moves, Google Fit	iHealth BG5	iHealth BP5	eMotion Faros
d*****s	Medisana BS 440	Withings, Moves, Google Fit	Medisana MediTouch2	Medisana BU575	-
d*****j	MedisanaTarget	iHealth, Moves, Google Fit	-	-	eMotion Faros
a*****l	Withings Scales	Fitbit Flex, Moves, Google Fit	-	Withings Blood Pressure	eMotion Faros
s*****d	iHealth HS5	Medisana VIFIT, Moves, Google Fit	-	-	-
a*****s	-	Moves, Google Fit	-	-	-
a*****m	Fitbit Aria	Fitbit One, Moves, Google Fit	-	-	-
a*****t	-	Fitbit One, Moves, Google Fit	-	-	-
j*****p	-	-	Medisana MediTouch2	-	-
d*****s	-	Fitbit One	-	-	-

Table 6 shows SPARQL query used for calculation of the total number of triples in CARRE repository.

Table 6. SPARQL query for counting the number of triples stored in CARRE RDF repository

```
SELECT (count(*) as ?count) WHERE {
  ?s ?p ?o .
}
2247812 triples (as for 22/04/2015)
```

Graph – level overview (graph names and number of triples in each graph) can be obtained by using SPARQL query (Table 7) executed in the CARRE SPARQL end–point.

Table 7. SPARQL query to get the graph – level overview of data stored in CARRE Private RDF repository

```
SELECT ?graphName count(*) as ?cnt
WHERE {
  GRAPH ?graphName {
    ?s ?p ?o
  }
}
group by ?graphName
order by DESC(?cnt)
```

Table 8 shows the number of data triples contained in CARRE personalised graphs.

Table 8. The graphs in CARRE Private RDF repository and associated numbers of data triples

Graph name	No of data triples in the repository
https://carre.kmi.open.ac.uk/users/athird	1315996
https://carre.kmi.open.ac.uk/users/domantas	179333
https://carre.kmi.open.ac.uk/users/baprice	169262
https://carre.kmi.open.ac.uk/users/vaimaro	154333
https://carre.kmi.open.ac.uk/users/daista	87391
https://carre.kmi.open.ac.uk/users/rugodzinski	60717
https://carre.kmi.open.ac.uk/users/JPiwinski	51976
https://carre.kmi.open.ac.uk/users/Alukos	29586
https://carre.kmi.open.ac.uk/users/gkotsis	22350
https://carre.kmi.open.ac.uk/users/a951753m	17197
https://carre.kmi.open.ac.uk/users/jdomingue	11939
https://carre.kmi.open.ac.uk/users/Saulis	6825
https://carre.kmi.open.ac.uk/users/drosatosgr	2640
https://carre.kmi.open.ac.uk/users/darjege	592
https://carre.kmi.open.ac.uk/users/Andrius	137
https://carre.kmi.open.ac.uk/users/Mindaugasu	19
https://carre.kmi.open.ac.uk/users/nporto	17
https://carre.kmi.open.ac.uk/users/DamianD	17
https://carre.kmi.open.ac.uk/users/romask	6
https://carre.kmi.open.ac.uk/users/weihui	6

4.3. Summary of testing results

Functional and non-functional tests of the CARRE sensor aggregators shows that the implemented aggregators provide the required functionality:

- They integrate 3rd party vendors' clouds;
- They fetch data about the specific users;
- And they store this data into the private repository.

All API functions and SPARQL queries work as intended. From 10 test scenarios and 69 cases considered (Table 3 and Table 4), all 69 test cases passed successfully. The updated source code of the sensor aggregator is available at: https://www.carre-project.eu/innovation/sensor-aggregator/ and the web application of the sensor aggregator itself is available at: https://carre.kmi.open.ac.uk/devices/.

5. Testing of Aggregators for Personal Medical & Lifestyle Data

5.1. Personal Health Records Aggregator

Personal Health Records aggregator is to: 1) fetch the list of data understood and needed by the CARRE repository, 2) collect data from the configured PHR sources, 3) unify and merge the data as configured and, 4) post it to CARRE endpoint.

Current design approach of the PHR aggregator is modular and highly extendible via pluggable program behaviour modifying modules (i.e. plugins). Therefore mostly unit tests are used (i.e. testing the algorithm flow of the specific *unit* of the system, e.g. whether the specific plugin group is loaded upon program execution and called in a predefined manner), with a few functional ones testing larger parts of the system.

As the results provided by the system greatly depend on the implementation of specific plugins (e.g. different PHR input plugins, different data output format plugins, different data merging plugins), whose implementations and requirements could be subject to change. Therefore, their specific implementation test results are not included, but the general testing guidelines of each plugin group are.

Because of the reason outlined above, mock data adhering to the expected data structure and containing values, which result in predictable output, was used for most tests. For example, unit test testing whether the configuration loading mechanism works would use a mock configurator plugin, which would return predefined, hard-coded configuration adhering to Configuration data type – and we would assert whether the configuration returned is equal to hard-coded in a mock configurator plugin.

5.1.1. Specific testing plan and results

Table 9 presents scenarios and cases for functional testing of Personal Health Record Aggregator.

Table 9. Test scenarios and test cases of functional testing of Personal Health Record Aggregator

No	Goal	Test Scenario	Test Case	Results and Comments
T-01	Assert that available observables are retrieved from CARRE repository	Load CARRE configurator, check configuration data units.	CARRE repository.	Pass.
T-02	Assert that PHR sources provide relevant data.	Load test configuration, run PHR data retrieval, check results.	Mock data.	Pass with mock data.
T-03	Assert that data retrieved from two or more PHRs is merged correctly.	Load test configuration, retrieve test PHR data, load test merge plugin, merge data, check results.	Mock data.	Pass with mock data.
T-04	Assert that resource unification module validates data according to configuration.	Load test configuration, retrieve mock PHR data with data not specified in test configuration, check end-data.	Mock data	Pass with mock data.
T-05	Assert that output module exports data in correct format.	Load test configuration, retrieve test PHR data, merge data, process it through output module, check results.	Mock data	Pass with mock data.

Table 10 presents Unit testing results of PHR aggregator.

Table 10. Unit testing of Personal Health Record Aggregator

No.	Unit	Unit test	Test Case	Results and Comments
U-01	Configuration	UC-01	Load configurator plugins.	Pass with mock data.
		UC-02	Configuration is returned by configurator plugins.	Pass with mock data.
U-02	PHR source	US-01	Load PHR source provider plugins.	Pass with mock data.
		US-02	PHR source provider plugins return PHR data.	Pass with mock data.
		US-03	PHR source provider plugins adhere to configuration provided by the configurator.	Pass with mock data
U-03	Resource unification module	UR-01	Module loads configurators.	Pass
		UR-02	Module retrieves configuration.	Pass
		UR-03	Module loads PHR source providers.	Pass
		UR-04	Module calls PHR source providers with configuration.	Pass
		UR-05	Module loads data merge plugins.	Pass
		UR-06	Module merges data using merge plugins.	Pass with mock data.
		UR-07	Module loads output provider plugins.	Pass with mock data.
		UR-08	Module calls output provider plugins with merged data.	Pass with mock data.
U-04	Merger	UM-01	Load merge provider plugins.	Pass with mock data.
		UM-02	Merge provider plugins return merged data.	Pass with mock data.
U-05	Output	UO-01	Load output provider plugins.	Pass with mock data.
		UO-02	Output provider plugins return success state of the output.	Pass with mock data.

Table 11 shows tests for PHR aggregator software plugins that might be developed in the future by third parties in order to accommodate other PHR systems currently not included.

Table 11. Tests for Personal Health Record Aggregator plugins

Plugin	Test case	Test	Result	
			Pass	Fail
Configurator	TC-01	Configurator plugin implements Configurator Interface.	Configurator plugin can be used by the aggregator.	Configurator plugin can not be used by the aggregator.
	TC-02	Configurator returns expected configuration: data units.	Aggregator will return expected data units.	Aggregator will not return expected data units.
	TC-03	Configurator returns expected configuration: data sources.	Aggregator will return data from these data sources.	Aggregator will not return data from these data sources.
PHR source	TS-01	PHR source plugin implements PhrSource interface.	PHR source plugin can be used by the aggregator.	PHR source plugin can not be used by the aggregator.
	TS-02	PHR source plugin gets data from its data source.	Aggregator will return data from the data source.	Aggregator will not return data from the data source.
	TS-03	PHR source plugin returns data according to the data units provided by configuration.	Aggregator will not have to process and filter extraneous data from the data source.	Aggregator will have to process and filter extraneous data from the data source.
Merger	TM-01	Merger plugin implements Merger interface.	Merger can be used by the aggregator.	Merger can not be used by the aggregator.
	TM-02	Merger merges data according to its specification.	Aggregator will return correctly-merged data.	Aggregator will not return correctly-merged data.
Output	TO-01	Output plugin implements Output interface.	Output can be used by the aggregator.	Output can not be used by the aggregator.
	TO-02	Output plugin outputs data according to its specification.	Aggregator output will be as expected.	Aggregator output will not be as expected.

5.1.2. Data and meta-data populated in private repository

Data and meta-data stored in private repository greatly depends on the implementation of different pluggable modules of the aggregator, but the aggregator itself should be able to function only having a rudimentary master patient index containing the GUIDs of patients to be retrieved and their mappings to the identifiers in other PHR systems. No sensitive, private or personal data is stored in aggregators' private repository.

5.1.3. Summary of testing results

Unit tests and functional tests provide proof that the overall architecture of the aggregator works as expected, but there is still a significant part of code of concrete implementations of the plugins not covered by tests. This technical debt is to be eliminated in parallel to the process of fine-tuning the aggregator in the production environment.

5.2. Web Lifestyle Data Aggregator

5.2.1. Specific testing plan and results

The testing plan addresses each one of the aggregator components separately. Thus, Table 12, shows the test scenarios and cases for the functional testing of (a) Query Detector, addressing its versions intended for different browsers: Firefox browser, Chrome browser and (b) User Intention Extractor. Table 13, shows the test scenarios and cases of non-functional testing of Web Lifestyle Data Aggregator components. The aggregator was tested by 5 volunteers part of the project team. All participants filled in Participant Consent Form (see Annex 1).

Table 12. Test scenarios and test cases of functional testing of Web Lifestyle Data Aggregator.

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
Component: **Query Detector for Firefox browser**				
TS_F_L_01	Installation	TC-01	Download and install add-on (*.xpi) from a web page	Pass
		TC-02	Drag and drop add-on (*.xpi) as file to the area of browser	Pass
		TC-03	Install add-on from the Firefox marketplace	Pass
TS_F_L_02	Usage	TC-01	Search on Google webpage	Pass
		TC-02	Search on Google from Firefox search box	Pass
		TC-03	Search on Bing webpage	Pass
		TC-04	Search on Bing from Firefox search box	Pass
		TC-05	Search on Yahoo webpage	Pass
		TC-06	Search on Yahoo from Firefox search box	Pass
Component: **Query Detector for Chrome browser**				
TS_F_L_03	Installation	TC-01	Download and install extension (*.crx) from a web page	Pass (until Chrome v32.x)
		TC-02	Drag and drop extension (*.crx) as file to the area of browser	Pass (until Chrome v32.x)
		TC-03	Load unpacked extension in developer mode	Pass
		TC-04	Install extension from the Chrome Web Store	Pass
TS_F_L_04	Usage	TC-01	Search on Google webpage	Pass
		TC-02	Search on Google from Chrome navigation bar	Pass

		TC-03	Search on Bing webpage	Pass
		TC-04	Search on Bing from Chrome navigation bar	Pass
		TC-05	Search on Yahoo webpage	Pass
		TC-06	Search on Yahoo from Chrome navigation bar	Pass
Component: **User Intention Extractor**				
TS_F_L_05	Installation	TC-01	Windows: Install using the automatic setup	Pass
		TC-02	Windows: Install manually using the guidelines	Pass
		TC-03	Linux: Install manually using the guidelines	Pass
		TC-04	Mac: Install manually using the guidelines	Pass
TS_F_L_06	Set CARRE Account	TC-01	Insert normal credential information	Pass
		TC-02	Cancel 'Set Account' dialog	Pass
		TC-03	Provide insufficient credential information	Pass
		TC-04	Save & Validate account	Pass
		TC-05	Prompt for new user registration	Pass
TS_F_L_07	Launch User Intention Extractor on startup	TC-01	Register process to run on startup via windows automatic setup	Pass
		TC-02	Enable/Disable to run on startup via the system tray	Fixed
TS_F_L_08	Visualization of Results	TC-01	Change dates and show the resulted intentions	Pass
		TC-02	Refresh the resulted user intentions	Pass
		TC-03	Drag and drop nodes from the shown graph	Pass
		TC-04	Change the shown top-n [1-3] classified intentions	Pass
TS_F_L_09	Browse Database	TC-01	Navigate on database structure	Pass
		TC-02	Navigate on browse data	Pass
		TC-03	Browse the collected queries	Pass
		TC-04	Browse the classified intentions per query	Pass
		TC-05	Browse the available categories of intentions	Pass

Table 13. Test scenarios and test cases of non-functional testing of Web Lifestyle Data Aggregator.

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
Component: **Query Detector for Firefox browser**				
TS_N_I_0	Integration	TC-01	User Intention Extractor	Pass

		TC-02	Firefox History Observer	Pass
TS_N_L_02	Data validation	TC-01	Pass all the queries to User Intention Extractor	Pass
		TC-02	Integrity of transmitted data	Pass
		TC-03	Check for multi-records of the same query	Fixed
		TC-04	Check for unsupported characters in queries	Pass
TS_N_L_03	Security	TC-01	Network transmission leakages over Internet	Pass
		TC-02	Violation of Firefox Security	Pass
Component: **Query Detector for Chrome browser**				
TS_N_L_04	Integration	TC-01	User Intention Extractor	Pass
		TC-02	Chrome History Listener	Pass
TS_N_L_05	Data validation	TC-01	Pass all the queries to User Intention Extractor	Pass
		TC-02	Integrity of transmitted data	Pass
		TC-03	Check for multi-records of the same query	Fixed
		TC-04	Check for unsupported characters in queries	Pass
TS_N_L_06	Security	TC-01	Network transmission leakages over Internet	Pass
		TC-02	Violation of Chrome Security	Pass
Component: **User Intention Extractor**				
TS_N_L_07	CARRE Private RDF – SPARQL Endpoint	TC-01	Integrity of transmitted data	Pass
		TC-02	SPARQL query validation	Pass
		TC-03	Check the correctness of inserted data types	Pass
		TC-04	Overcome problems of connection lost	Pass
TS_N_L_08	Data validation	TC-01	Receive all queries from Query Detectors	Pass
		TC-02	Pass all queries to classification process	Pass
		TC-03	Store all detected intentions to the local database of user	Pass
		TC-04	Upload the all relevant intentions of user private RDF	Pass
TS_N_L_09	Security	TC-01	Network transmission leakages over Internet	Fixed
		TC-02	Safe storage of user credentials	Pass

		TC-03	Not upload the actual queries to private RDF	Pass
		TC-04	Not upload any data to other RDF repositories apart from the user private RDF	Pass
		TC-05	Only health and travelling intentions upload to CARRE private RDF	Pass

5.2.2. Data and meta-data populated in private repository

Table 14, shows the web Lifestyle data that is collected by the Web Lifestyle Data Aggregator for 5 volunteers. All participating volunteers created private accounts in CARRE system and installed the required components (Query Detector for Firefox and Chrome, and User Intention Extractor) of aggregator in their personal computers. The only thing that they had to do was to search on the Internet as they do every day. The numbers that are shown in Table 14 represent the search queries (light blue) and the relevant intentions (light orange) of users for a period of 1 month (20/03/2015 - 20/04/2015). In order to fill the Table 14, we added an additional functionality in the User Intention Extractor, which is responsible to provide statistical information about the collected data to the user. Note that the actual queries are only stored in a local database that is located in user's personal computer and only the intentions that are relevant to CARRE are uploaded in the private RDF repository. The number of relevant intentions appears bigger than the number of relevant queries because we classify every query in the top-3 intentions according the available categories. Whether we want to transform the number of intentions to number of triples in RDF, we would only need to multiply the number of intentions with 7 (number of triples per intention). Finally, the usernames of volunteers are anonymized for privacy reasons.

Table 14. Web lifestyle related data and metadata populated in user local database and in private RDF.

Anonymized Username	Search Engines *(# Queries)*			Browsers *(# Queries)*		User Local Database		Private RDF
	Google	Bing	Yahoo	Firefox	Chrome	# Total Queries	# Relevant Queries	# Relevant Intentions
d********r	303	24	4	319	12	331	44	59
d****s	56	40	0	58	38	96	11	15
a****d	540	10	1	439	112	551	153	189
e*****o	24	4	7	15	20	35	2	3
n****o	729	0	0	10	719	729	36	42

5.2.3. Summary of testing results

From the 68 test scenarios and cases considered (Table 12 and Table 13) 62 passed successfully. From the 6 test cases with issues, the 2 tests (TS_F_L_03.TC-01 and TS_F_L_03.TC-02) conditionally passed and the 4 tests (TS_N_L_02.TC-03, TS_N_L_05.TC-03, TS_F_L_07.TC-02 and TS_N_L_09.TC-01) fixed. Table 15 summarizes the test cases that revealed issues and bugs of the component and the corrective measures taken.

Table 15. Discussion and solutions about the test cases with issues.

Query Detector for Firefox browser	
TS_N_L_02.TC-03	Details: We check if the Query Detector detects a search query more than one time. This depends on how the webpage of search engine works. Results: We determine that when a user clicks the next pages of results the Query Detector detects again the same query. We fixed this issue and we release a new version (v0.4) of add-on.
Query Detector for Chrome browser	
TS_F_L_03.TC-01	Details: We test if it possible to install someone the Chrome extension of Query Detector that is available to the CARRE webpage. Therefore, the tester downloads the CRX file and installs it to Chrome with a simple click on the file. Results: We do not find any problem with the following procedure until the Chrome v32.x, after this version there are some security restrictions of Chrome browser that allow installing an extension only from the Chrome Web Store.
TS_F_L_03.TC-02	Details: We test if it possible to install someone the Chrome extension of Query Detector with a simple drag and drop of the CRX file to the Chrome window. Results: We do not find any problem with the following procedure until the Chrome v32.x, after this version there are some security restrictions of Chrome browser that allow installing an extension only from the Chrome Web Store.
TS_N_L_05.TC-03	Details: We check if the Query Detector detects a search query more than one time. This depends on how the webpage of search engine works. Results: We determine that when a user clicks the next pages of results the Query Detector detects again the same query. We fixed this issue and we release a new version (v0.3) of extension.
User Intention Extractor	
TS_F_L_07.TC-02	Details: In the menu of User Intention Extractor, the user is possible to select if he/she wants to enable or disable the running of this component on the startup of the operating system. Results: This feature was not fully functional and by default the User Intention Extractor has defined to run on the startup. We fixed this issue and we release a new version (v1.3) of the application.
TS_N_L_09.TC-01	Details: We perform a series of networking tests (using as packet analyzer the Wireshark[13]) in order to see what data is transmitted in plaintext form or not and in which direction. Results: The results of this procedure show that the communication channel between the User Intention Extractor and the authenticated SPARQL endpoint was unsecured (HTTP) by default. Thus, we fixed this issue by passing this channel via a secure communication channel (HTTPS) that is provided by the Apache web server (mod_proxy[14]).

[13] https://www.wireshark.org/

[14] http://httpd.apache.org/docs/current/mod/mod_proxy.html

6. Testing of Aggregators for Medical Scientific and Educational Data

6.1. Medical Evidence Data Aggregator

The medical evidence data aggregator aims to: 1) enrich the evidence of the existing risk descriptions; and 2) identify new risk associations for cardiorenal diseases and comorbidities as published in medical literature during and beyond the project's lifetime. Medical experts can use key words to search, then based on the automatically highlighted new evidences can suggest possible new risk associations, and analyse and confirm the mined evidences and associations.

6.1.1. Specific testing plan and results

The test addresses functional and non-functional issues for the portal. Table 16 shows test scenarios and test cases of Functional testing.

Table 16. Test scenarios and test cases of functional testing of Medical Evidence Data Aggregator.

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_F_E_01	Data search from data source (search interface by keywords to get list)	TC-01	Pubmed data source	Pass
		TC-02	IEEE data source	Pass
TS_F_E_02	Data fetch from data source (fetch data from id to get details)	TC-01	Pubmed data source	Pass
		TC-02	IEEE data source	Pass
TS_F_E_03	Interface for Data search, fetch, list, refine, paging	TC-01	Data pass to search engine and return summary: user give a keyword and show the list	Pass
		TC-02	Data pass to search engine and fetch abstract: user double click a citation and fetch the abstract	Pass
		TC-03	Order by: list interface interaction: the order of list can be organized by summary year and flag	Pass
		TC-04	Filter from the exist list: only show items that contains the given string	Pass
		TC-05	Previous / next page: interface interaction	Pass with default 20 records
TS_F_E_04	Dictionary organization	TC-01	Load user specified dictionaries: initialize selected dictionary for analysis	Pass with CARRE, ICD,HOTMAP,CL
		TC-02	Drag/drop to manage selected/unselected dictionary: add/insert/delete dictionary from a list	Pass

TS_F_E_05	Timing task-pick up/collect citations	TC-01	Pick up a task at specified time every day: pick up a list of key words of a domain	Pass
		TC-02	Read the task and collect 20 citations for each keyword: document repository growing regularly by the task	Pass
		TC-03	Admin user could set task/processing task: task be loaded to data repository and task be processed	Pass
TS_F_E_06	Statistics	TC-01	Statistic new pairs: show all the documents that contains new pairs	Pass with a small set of Risk/Result
		TC-02	It should only contain specified domain not others	Pass
		TC-03	Risk and result be picked out and their relations	Pass
		TC-04	Interface interaction: expand/collapse, terms in right colour	Pass
TS_F_E_07	Medical Evidence Data	TC-01	Citation showed in right format: sectioned (introduction, method, results and conclusion) or not (in whole paragraph)	Pass, tested two format citations
		TC-02	Input analysis keywords: popup window to allow user input a group of keywords	Pass
		TC-03	Keyword tags create: user keywords tags should be created after analysis done	Pass
		TC-04	Basic tags: number tags should be created from configure file	Pass
		TC-05	Project tags: project tags should be created from configure file	Pass
		TC-06	Relation table: output including a relation table if there are some relations	Pass, need improvement
		TC-07	Tag interface: check/uncheck one check box should fire an event if there is.	Pass
		TC-08	Check/uncheck all tags should fire some events if there is.	Pass
		TC-09	Keywords tags: if the citation contains the keywords user defined, this/these keyword(s) should be picked out by high light colour.	Pass
		TC-10	Risk tags: if a risk term was contained in the citation, and this term was listed in the ontology, this term should be picked out by a high light colour.	Pass if it is defined by ontology
		TC-11	Result tags: if a result term was contained in the citation, and this term was listed in the ontology, this term	Pass if it is defined by ontology

			should be picked out by a high light colour.	
		TC-12	Negative tags: if a negative term was contained in the citation, and this term was listed in the ontology, this term should be picked out by a high light colour.	Pass if it is defined by ontology
		TC-13	Positive Strong tags: if risk and result terms contained in one sentence and they have been defined as cause/caused by relation, this sentence should be picked out by a high light colour.	Pass, needs improvement
		TC-14	Positive Weak tags: if risk and result terms contained in one sentence but they have not been defined as cause/caused by relation, this sentence should be picked out by a high light colour.	Pass
		TC-15	Negative strong tags: if risk and result terms contained in one sentence, they have been defined as cause/caused by relation, but this sentence contains negative tags, this sentence should be picked out by a high light colour.	Pass, needs improvement
		TC-16	New pair tags: if one result tag in a sentence, the grammar structure of this sentence is clear, the corresponding triple should be picked out and listed in the new pair section.	Pass, needs improvement
		TC-17	Unknown knowledge: if a sentence contains some terms belong to risk and result tags, but they do not have relation defined, they will be listed in unknown knowledge section.	Pass
TS_F_E_08	Document access	TC-01	findDocument - Return document by given docId	Pass
		TC-02	getAttachmentsByDocId - Return a list including all the attachments that included by this doc	Pass
		TC-03	getDBUri - Return URI of this document repository	Pass
		TC-04	removeDocument - Remove given document	Pass
		TC-05	SaveDocument - Create or update a document by given model	Pass
		TC-06	View - Return document list by given view Id	Pass
TS_F_	PaperDocAccess	TC-01	addAttachmentForPaper - add a Attachment	Pass

		TC-02	addHtmlAttachmentForPaper - add a Html Attachment	Pass
		TC-03	addXmlAttachmentForPaper - add a Xml Attachment	Pass
		TC-04	GetDocAttachemtConten - getDocAttachemt by id and name	Pass
		TC-05	getPaperDocumentByDocId - get Paper Document By DocId	Pass
		TC-06	getPaperDocumentByResIdAndResType - get Paper Document By ResId And ResType	Pass
		TC-07	getPaperIdsByAuthor - get Paper Ids By Author	Pass
		TC-08	getPaperIdsByEndYear - get Paper Ids By End Year	Pass
		TC-09	getPaperIdsByPubTitle - get Paper Ids By PubTitle	Pass
		TC-10	getPaperIdsByYear - get PaperIds By Year	Pass
		TC-11	getPaperIdsByYearAndAuthor - get PaperIds By Year And Author	Pass
		TC-12	getPaperIdsByYearAndPubTitle - get PaperIds By Year And PubTitle	Pass
		TC-13	getPaperIdsByYearAndTerm - get Paper Ids By Year And Term	Pass
		TC-14	getPaperIdsByYearRange - get Paper Ids By Year Range	Pass
		TC-15	getPaperTermsByDocId - get Paper Terms By DocId	Pass

Table 17 shows test scenarios and test cases of Non-Functional testing.

Table 17. Test scenarios and test cases of non-functional testing of Medical Evidence Data Aggregator

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
TS_N_E_01	System initialize	TC-01	Gate initialize: Gate home found, plugins loaded	Pass
		TC-02	System configure load: System properties load	Pass
		TC-03	System resource load: Jape files	Pass
		TC-04	Ontology files	Pass
		TC-05	Domain keywords	Pass
		TC-06	RDF repository connect: Get response from repository	Pass

TS_N_E_02	Timing task	TC-01	Pickup task start-up: Load admin user uploaded domain keywords	Pass with a configured time
		TC-02	Process task start-up: Collect document	Pass
TS_N_E_03	Citation processing init	TC-01	clearNLP init properly - Decoder get instance	Pass
		TC-02	Pattern files load - Sentence pattern resource load	Pass
TS_N_E_04	Key words rule	TC-01	Rule create - Temp rule created	Pass
TS_N_E_05	Building processing pipeline, user defined dictionaries and user defined keywords	TC-01	Build basic pipelines - No exception	Pass
		TC-2	Build ontology for each dictionary	Pass with CARRE, ICD,HOTMAP,CL
		TC-3	Build JAPE transducer for keywords	Pass with fixed format
TS_N_E_06	Analysis semantic role	TC-1	Pick out sentences containing feature: analysis verb action for these sentences	Pass with blank feature

6.1.2. Data and meta-data populated in public repository

The RDF repository was designed to store the medical evidence data, described in deliverable D4.1 "Semantic repository design & implementation". The original data were gathered as a result of the manual search for relevant medical evidence data as defined in deliverable D2.2 "Functional Requirements & CARRE Information Model". The results of task T3.4 "Aggregators for medical evidence" provide ways for automatic search and suggest for possible new risk associations, which need further validation from the medical professionals. Once the new risk associations have been validated, CARRE experts are switching back to the Semantic Data Entry System in order to insert the data.

6.1.3. Summary of testing results

We have created 9 function scenarios of 56 test cases with 46 clean pass and 6 non functional scenarios of 15 test cases based on current progress. Table 18 summarizes the test cases with issues which are not clean pass or under condition.

Table 18. Discussion and solutions about the test cases with issues.

TS_F_E_03.TC-05	Issue: total number of displayed items can be changed
TS_F_E_04.TC-01	Issue: currently only limited ontology had been tested, such as ICD, CL, CARRE, other tests can be conducted if needed.
TS_F_E_06.TC-01	Issue: need to test on performance when dataset grow larger. Need to test with Risk/Result ontology from OU.
TS_F_E_07.TC-01	Issue: we found two formats in PubMed, there are maybe more.
TS_F_E_07.TC-06	Issue: words relation in simple sentence structures were tested.
TS_F_E_07.TC-10 to TC-13	Issue: if format is not standard, the accuracy will generally decrease. Ontology should improve the accuracy, for example use synonyms.

TS_F_E_07.TC-15 to 16	Issue: could improve it by checking close relation and use ontology defined by OU.
TS_N_E_02.TC-1	Issue: time zone could change when needed. .Have not tested cases when server was migrated to other time zone server.
TS_N_E_05.TC-2	Issue: if need more we have to test it first, especially when the dictionary is too big.
TS_N_E_05.TC-03	If format needs to be changed, code needs to be modified as well. Currently sample format is used.
TS_N_E_06.TC-01	Issue: needs to find featured sentence by using latest CARRE ontology, and then to analysis sentence structures.

New release of developed software (open source) source code can be found here:

http://www.carre-project.eu/innovation/medical-evidence-aggregator/

6.2. Educational Resource Aggregator

In this section, we will describe the testing methods and results for the Educational Resource Aggregator.

The testing is done per module basis and will be further divided in Functional and non-Functional. Furthermore, we will briefly describe a group of software automated unit testing that is integrated on the development process of each component and provides data integrity and validation checks during development stages. Finally, in the end of this section we will report a summary of what we encountered and fixed during the integration and testing process.

In order for the reader to follow along the above statements, a brief description is provided from deliverable T3.4.

The aim of the educational resource aggregator is to harvest educational resources from 3rd party repositories, present these to the medical expert for annotation and rating, and output the results of the annotation (together with resource metadata) to the CARRE public RDF repository.

The main parts of this aggregator are: the **Resource Retriever**, the **Resource Rating**, the **Resource Metadata Processing**, and the **User Application**.

- The **Resource Retriever** accepts CARRE concept terms from the CARRE public RDF repository and uses them to formulate queries to external 3rd party educational resource repositories. The results of this search are parsed to extract metadata. Then the retrieved results and metadata are displayed to the expert user for rating and annotation (via the aggregator front end). The module consists of 2 services that make use of SPARQL protocol in the case of query term extraction from the CARRE server and API requests to each educational repository.

- The **Resource Rating** module allows the input of expert user opinion and annotation, and also calculates subjective scores that measure the quality of the resource. Expert rating involves assessment of content-keyword relevance, content accuracy and depth of coverage, while the automatic systems rating is based on the Readability Test of the Flesch-Kincaid algorithm, and rating based on the latest modified version of the article and number of revisions.

- The **Resource Metadata Processing** module involves metadata enrichment via semantic web sources (such as NCBO BioPortal medical ontologies and DBpedia). The module is a combination of 3 services that collect data per article, making multiple SPARQL requests to enrich the data and finally store it as a unique identified resource into the local MongoDB datastore. Then data is transformed into RDF triples in order to be inserted to CARRE educational repository.

- The **User Application** is a web application based on HTML5 technology with responsive views for almost every device available, mobile phones , tablet , desktop pc's and embedded industrial pc's. The user applications integrates the above modules and enables user interaction with the data components

6.2.1. Specific testing plan and results

The testing plan as explained in introduction is organized by modules according to the Architecture of the Educational Resources Aggregator described in T3.4. The numbering of the test scenarios follows the “TS_F_ED_” naming convention. In detail, both the functional testing and non-functional is further divided in two modules, backend interface and frontend user interface testing.

The functional testing is analytically described in Table 19 and involves scenarios about functionality review and design validation regarding each component of the Educational Resource Aggregator.

The non-functional testing is shown in Table 20 and addresses issues about integration testing, data validation, scalability testing, browser and compatibility testing, security testing and performance testing. Moreover, we will not investigate scenarios about API testing, because the aggregator does not expose any API thus no testing is available. In the last subsection, we will describe in detail all these test cases that have issues and how we resolved them.

In the last section a brief description takes place about automated testing during development and some screenshots of Travis CI[15] and other tools used to enhance the development cycle.

Table 19. Test scenarios and test cases of functional testing of Educational Resource Aggregator.

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
Module : Backend interface of Educational Resource Aggregator				
TS_F_ED_01	Fetch CARRE risk elements	TC01	Establish backend connection with public RDF repository by educationalAggregator user	Pass
		TC02	Fetch all CARRE risk elements from public RDF repository	Pass
		TC03	Temporarily store to Local Storage for better performance	Not implemented as of V0.4.4[16]
TS_F_ED_02	Query Generator	TC01	Formulate queries for the unstructured web repositories	Pass
		TC02	Query to Wikipedia in JSON format	Pass
		TC03	Query to MedlinePLUS in XML format	Pass
		TC04	Parse results from Wikipedia and MedlinePLUS and unify data into JS objects	Pass
TS_F_E		TC01	Retrieve JSON list from Wikipedia	Pass

[15] Travis CI is an open-source hosted, distributed continuous integration service used to build and test projects hosted at GitHub, https://travis-ci.org/

[16] the temporary storage functionality had been used in early versions for testing – no longer applicable

	Educational Object Harvester	TC02	Retrieve resource from MedlinePlus	Pass
		TC03	Filter out resources unrelated to Health or CARRE model	Pass
TS_F_ED_04	Educational Object Metadata Extractor	TC01	Extract Wikipedia resource snippet and source URL	Pass
		TC02	Extract MedlinePLUS snippet and source URL	Pass
		TC03	Semantic Analysis of fetched Resource	Not yet as of V0.4.4
TS_F_ED_05	Metadata Enrichment and Mapping CARRE schema	TC01	Medical identifiers fetched for each resource visited by user	Not implemented as of V0.4.4
		TC02	Resource linked to CARRE risk elements	Fixed
TS_F_ED_06	Educational Metadata Sender	TC01	RDF insert queries per resource	Pass
TS_F_ED_07	Educational Object Rating	TC01	Retrieve ratings for all user rated resources that exist in current search results	Pass
TC02	RDF insert rating score per resource per user	Pass		
Module: Frontend user interface of Educational Resource Aggregator				
TS_F_ED_08	Login/Logout and Register through CARRE devices	TC01	Login/Logout redirection through https	Fixed
		TC02	Register through https	Fixed
TS_F_ED_09	Search for educational resources	TC01	Autocomplete risk elements through CARRE RDF repository	Pass
		TC02	Allow query without risk element	Pass
		TC03	Display current average rating of each resource	Pass

		TC04	Pagination when results are more than 20 on top and bottom of page	Pass
		TC05	Wikipedia/MedlinePLUS source select box	Pass
		TC06	Top search box functionality	Pass
		TC07	Default search engine set as Wikipedia	Pass
TS_F_ED_10	Navigate to educational resource	TC01	Click on resource navigates to Resource Details	Pass
		TC02	Back button returns you to previous resource list	Pass
		TC03	Resource showing in Iframe compatibility with source website	Fixed
TS_F_L_11	User Rating	TC01	Rating visible only for logged in users	Pass
		TC02	Special Rating for each user role	Not implemented as of V0.4.4
		TC03	Rating average score gets updated live after each user rating	Fixed in V0.2.3

Table 20. Test scenarios and test cases of non-functional testing of Educational Resource Aggregator.

Test Scenario		Test Case		Results and Comments
No	*Description*	*No*	*Description*	
Module : Backend interface of Educational Resource Aggregator				
TS_N_ED_01	Security	TC01	Get token for write access to public RDF through ENV variables	Pass
		TC02	Only logged in users allowed to modify/add ratings in RDF repository	Pass
		TC03	Transmit data through encrypted protocol (https)	Fixed
TS_N_ED_02	Data security testing	TC01	Duplicate educational resource are not allowed	Pass
		TC02	Users cannot add duplicate ratings on each resource	Pass
		TC03	Users can edit only their own rated resources	Pass
		TC04	User authentication saved temporarily in encrypted browser session	Fixed
TS_N_ED_03	Performance Testing	TC01	Retrieve resources from Wikipedia	Pass (200ms)
		TC02	Retrieve resources from MedlinePlus	Pass (450ms)

		TC03	Average query to RDF repository (Virtuoso)	Pass (250ms)
		TC01	Process resource metadata after retrieval	Pass (170ms)
		TC02	Upload resource metadata to RDF repository	Pass(220ms)
		TC03	Upload resource rating metadata to RDF	Pass(280ms)
Module : Frontend user interface of Educational Resource Aggregator				
TS_N_ED_0	Data type validation testing	TC01	Search query is validated against special characters	Pass
		TC02	Queries passed to online sources are URL encoded	Pass
TS_N_ED_05	Browser Compatibility	TC01	Chrome after >=30	Pass
		TC02	Firefox after >=12	Pass
		TC03	Opera after >=7	Pass
		TC04	Internet Explorer >=10	Pass
		TC05	Safari >=4.0	Pass
		TC06	Android >=4.0	Pass
		TC07	IOS >=5.0	Pass
TS_N_ED_06	Scalability Testing	TC01	Local Database scalability	Fixed

Scalability Testing

The previous implementation fails on scalability needs because of MongoDB data store unit, which is not scalable by design.The current implementation as of V0.4.4 removes the dependency of the above unit thus the only storage engine used is Virtuoso RDF repository. As a result the Educational Aggregator follows the 12-factor app principle.

The Twelve-Factor App is a methodology for building software-as-a-service apps that:

- Use declarative formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a clean contract with the underlying operating system, offering maximum portability between execution environments;
- Are suitable for deployment on modern cloud platforms, obviating the need for servers and systems administration;
- Minimize divergence between development and production, enabling continuous deployment for maximum agility;
- Can scale up without significant changes to tooling, architecture, or development practices.

The Twelve-Factor methodology can be applied to applications written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

In order to prove the previous statement, we provide a set of benchmarking tests using Apache Bench:

1. Results of 200 simultaneous users calling 1 instance of the aggregator on a set of 2000 requests

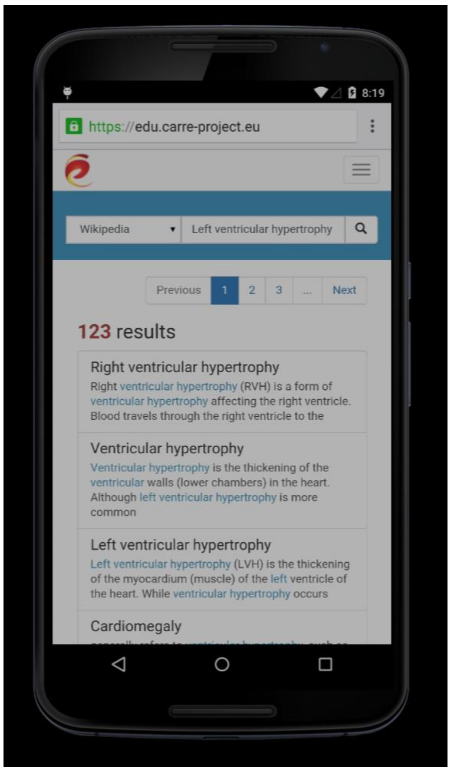
```
Server Software:        nginx/1.4.6
Server Hostname:       edu.carre-project.eu
Server Port:           80

Document Path:         /
Document Length:       1568 bytes

Concurrency Level:     200
Time taken for tests:  6.442 seconds
Complete requests:     2000
Failed requests:       0
Total transferred:     4079176 bytes
HTML transferred:      3136000 bytes
Requests per second:   310.47 [#/sec] (mean)
Time per request:      644.176 [ms] (mean)
Time per request:      3.221 [ms] (mean, across all concurrent requests)
Transfer rate:         618.40 [Kbytes/sec] received
```

2. Results of 400 simultaneous users calling 3 instances of the aggregator on a set of 2000 requests

```
Server Software:       nginx/1.4.6
Server Hostname:       edu.carre-project.eu
Server Port:           80

Document Path:         /
Document Length:       1568 bytes

Concurrency Level:     400
Time taken for tests:  4.354 seconds
Complete requests:     2000
Failed requests:       0
Total transferred:     4079176 bytes
HTML transferred:      3136000 bytes
Requests per second:   459.35 [#/sec] (mean)
Time per request:      355.756 [ms] (mean)
Time per request:      2.177 [ms] (mean, across all concurrent requests)
Transfer rate:         756.39 [Kbytes/sec] received
```

Browser Compatibility

Below we present sample screenshots of the browser compatibility testing for mobile, tablet and desktop views.



123 results

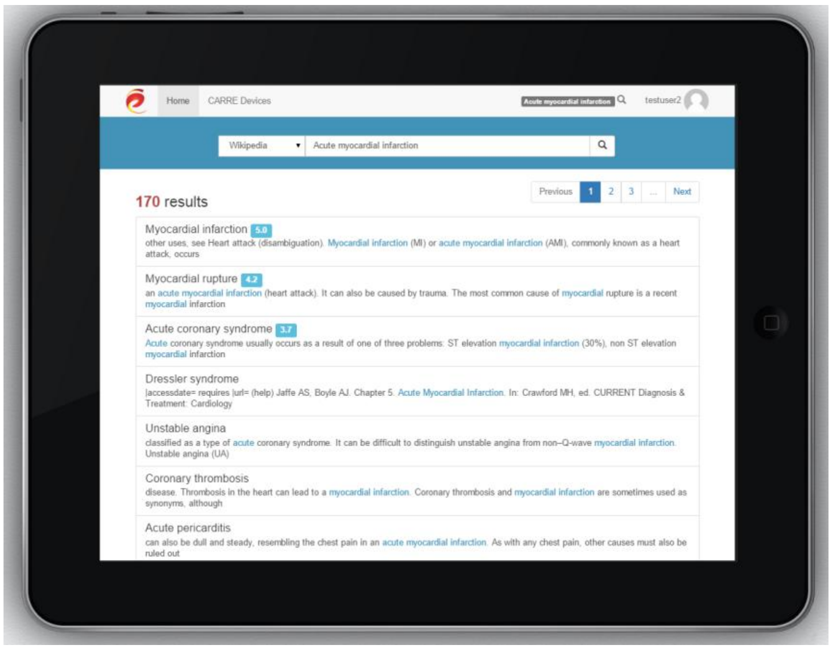
Right ventricular hypertrophy
Right ventricular hypertrophy (RVH) is a form of ventricular hypertrophy affecting the right ventricle. Blood travels through the right ventricle to the

Ventricular hypertrophy
Ventricular hypertrophy is the thickening of the ventricular walls (lower chambers) in the heart. Although left ventricular hypertrophy is more common

Left ventricular hypertrophy
Left ventricular hypertrophy (LVH) is the thickening of the myocardium (muscle) of the left ventricle of the heart. While ventricular hypertrophy occurs

Cardiomegaly

Figure 1. A screenshot of CARRE Educational Resources aggregator on Google Nexus 6



170 results

Myocardial infarction 5.0
other uses, see Heart attack (disambiguation). Myocardial infarction (MI) or acute myocardial infarction (AMI), commonly known as a heart attack, occurs

Myocardial rupture 4.2
an acute myocardial infarction (heart attack). It can also be caused by trauma. The most common cause of myocardial rupture is a recent myocardial infarction

Acute coronary syndrome 3.7
Acute coronary syndrome usually occurs as a result of one of three problems: ST elevation myocardial infarction (30%), non ST elevation myocardial infarction

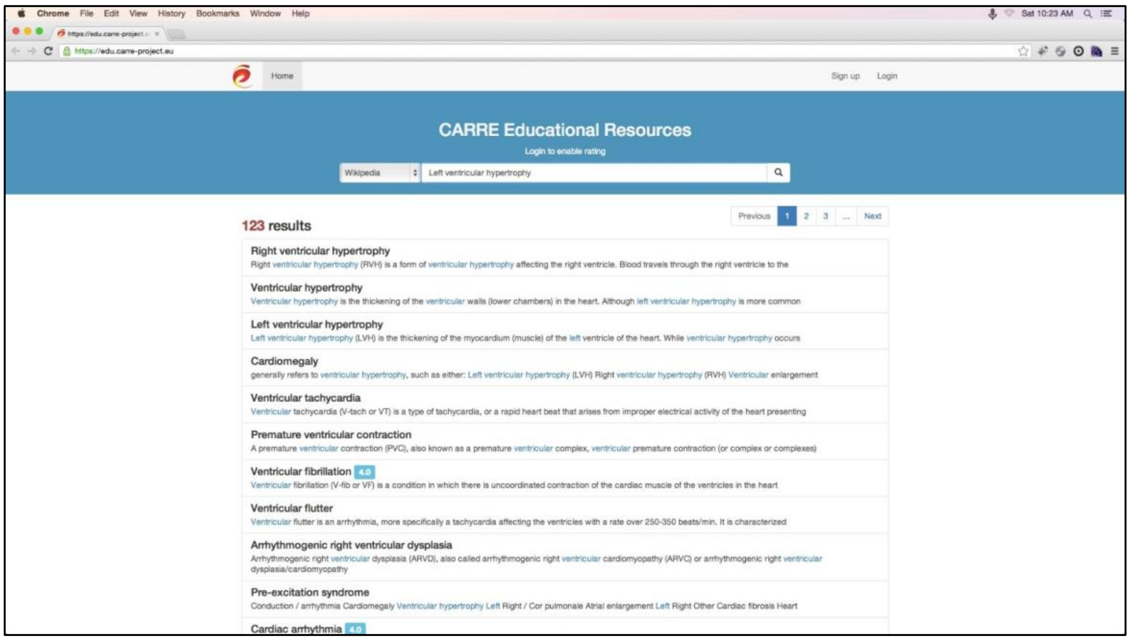
Dressler syndrome
|accessdate= requires |url= (help) Jaffe AS, Boyle AJ. Chapter 5. Acute Myocardial Infarction. In: Crawford MH, ed. CURRENT Diagnosis & Treatment: Cardiology

Unstable angina
classified as a type of acute coronary syndrome. It can be difficult to distinguish unstable angina from non–Q-wave myocardial infarction. Unstable angina (UA)

Coronary thrombosis
disease. Thrombosis in the heart can lead to a myocardial infarction. Coronary thrombosis and myocardial infarction are sometimes used as synonyms, although

Acute pericarditis
can also be dull and steady, resembling the chest pain in an acute myocardial infarction. As with any chest pain, other causes must also be ruled out

Figure 2. A screenshot of CARRE Educational Resources aggregator on iPad 4th gen


Chrome File Edit View History Bookmarks Window Help
Sat 10:23 AM
https://edu.carre-project.eu
Home
Sign up
Login
Wikipedia
Previous
1
2
3
...
Next

CARRE Educational Resources

Login to enable rating

Left ventricular hypertrophy

123 results

Right ventricular hypertrophy
Right ventricular hypertrophy (RVH) is a form of ventricular hypertrophy affecting the right ventricle. Blood travels through the right ventricle to the

Ventricular hypertrophy
Ventricular hypertrophy is the thickening of the ventricular walls (lower chambers) in the heart. Although left ventricular hypertrophy is more common

Left ventricular hypertrophy
Left ventricular hypertrophy (LVH) is the thickening of the myocardium (muscle) of the left ventricle of the heart. While ventricular hypertrophy occurs

Cardiomegaly
generally refers to ventricular hypertrophy, such as either: Left ventricular hypertrophy (LVH) Right ventricular hypertrophy (RVH) Ventricular enlargement

Ventricular tachycardia
Ventricular tachycardia (V-tach or VT) is a type of tachycardia, or a rapid heart beat that arises from improper electrical activity of the heart presenting

Premature ventricular contraction
A premature ventricular contraction (PVC), also known as a premature ventricular complex, ventricular premature contraction (or complex or complexes)

Ventricular fibrillation 4.0
Ventricular fibrillation (V-fib or VF) is a condition in which there is uncoordinated contraction of the cardiac muscle of the ventricles in the heart

Ventricular flutter
Ventricular flutter is an arrhythmia, more specifically a tachycardia affecting the ventricles with a rate over 250-350 beats/min. It is characterized

Arrhythmogenic right ventricular dysplasia
Arrhythmogenic right ventricular dysplasia (ARVD), also called arrhythmogenic right ventricular cardiomyopathy (ARVC) or arrhythmogenic right ventricular dysplasia/cardiomyopathy

Pre-excitation syndrome
Conduction / arrhythmia Cardiomegaly Ventricular hypertrophy Left Right / Cor pulmonale Atrial enlargement Left Right Other Cardiac fibrosis Heart

Cardiac arrhythmia 4.0

Figure 3. A screenshot of CARRE Educational Resources aggregator on Mac OS X – Chrome browser

6.2.2. Data and meta-data populated in public repository

In the current subsection, we describe queries and corresponding results for data populated in RDF repository. The queries have been requested to http://carre.kmi.open.ac.uk:8890/sparql (Public RDF Endpoint) in the default graph http://carre.kmi.open.ac.uk/public :

Table 21. SPARQL query to count all educational material per risk element

```
PREFIX edu: <http://carre.kmi.open.ac.uk/ontology/educational.owl#>
PREFIX risk: <http://carre.kmi.open.ac.uk/ontology/risk.owl#>
select distinct ?Risk_Element count(?title) AS ?Educational_Material WHERE {
?Risk_Element risk:has_educational_material ?p.
?p edu:title ?title.
}
GROUP BY ?Risk_Element
ORDER BY ?Educational_Material
```

Table 22. The SPARQL query results – the number of educational material per risk element

Risk Element	Educational Material
CKD stage (http://carre.kmi.open.ac.uk/elements/ckd_3b)	1
CKD stage 4 (http://carre.kmi.open.ac.uk/elements/ckd_stage4)	3
CKD progression (http://carre.kmi.open.ac.uk/elements/progression_of_ckd)	5
Smoking (http://carre.kmi.open.ac.uk/elements/smoking)	5
CKD stage 5 (http://carre.kmi.open.ac.uk/elements/ckd_stage_5)	8
CKD stage 3 (http://carre.kmi.open.ac.uk/elements/ckd_3a)	9

Acute kidney injury (http://carre.kmi.open.ac.uk/elements/acute_kidney_injury)	12
Atrial fibrillation (http://carre.kmi.open.ac.uk/elements/atrial_fibrillation)	12
Obesity (http://carre.kmi.open.ac.uk/elements/obesity)	14
Central obesity (http://carre.kmi.open.ac.uk/elements/central_obesity)	19
Dyslipidaemia (http://carre.kmi.open.ac.uk/elements/dyslipidemia)	19
Age (http://carre.kmi.open.ac.uk/elements/age)	19
Death (http://carre.kmi.open.ac.uk/elements/death)	20
Left ventricular hypertrophy (http://carre.kmi.open.ac.uk/elements/left_ventricular_hypertrophy)	21
Hypoglycaemia (http://carre.kmi.open.ac.uk/elements/hypoglycaemia)	21
Diabetes (http://carre.kmi.open.ac.uk/elements/diabetes)	22
Acute myocardial infarction (http://carre.kmi.open.ac.uk/elements/acute_myocardial_infarction)	27
Ischemic stroke (http://carre.kmi.open.ac.uk/elements/ischemic_stroke)	28
Depression (http://carre.kmi.open.ac.uk/elements/depression)	30
Cardiovascular disease (http://carre.kmi.open.ac.uk/elements/cardiovascular_disease)	32
Heart failure (http://carre.kmi.open.ac.uk/elements/heart_failure)	33
Ischemic heart disease (http://carre.kmi.open.ac.uk/elements/ischemic_heart_disease)	34
Hypertension (http://carre.kmi.open.ac.uk/elements/hypertension)	34
Chronic kidney disease (http://carre.kmi.open.ac.uk/elements/chronic_kidney_disease)	34
Anemia (http://carre.kmi.open.ac.uk/elements/anemia)	53

Table 23. SPARQL query to count all educational material for each web source

```
PREFIX edu: <http://carre.kmi.open.ac.uk/ontology/educational.owl#>
select ?Source count(?p) AS ?Educational_Material WHERE {
?p edu:websource ?Source.
}
GROUP BY ?Source
```

Table 24. The SPARQL query results – the number of educational material for each web source

Source	Educational_Material
Wikipedia	312
Medlineplus	127

Table 25. The SPARQL query to count all predicates from the educational resource ontology

```
SELECT ?Predicates count(*) as ?Count
WHERE {
  ?s ?Predicates ?o .
  FILTER regex(str(?Predicates), "^http://carre.kmi.open.ac.uk/ontology/educational.owl")
}
group by ?Predicates
order by DESC(?Count)
```

Table 26. The SPARQL query results – the number of properties of the MERA ontology[17]

Properties of MERA	Count
http://carre.kmi.open.ac.uk/ontology/educational.owl#total	994
http://carre.kmi.open.ac.uk/ontology/educational.owl#for_article	963
http://carre.kmi.open.ac.uk/ontology/educational.owl#date	963
http://carre.kmi.open.ac.uk/ontology/educational.owl#query	820
http://carre.kmi.open.ac.uk/ontology/educational.owl#position	820
http://carre.kmi.open.ac.uk/ontology/educational.owl#url	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#title	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#views	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#date_accepted	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#language	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#websource	439
http://carre.kmi.open.ac.uk/ontology/educational.owl#snippet	438
http://carre.kmi.open.ac.uk/ontology/educational.owl#wordcount	308
http://carre.kmi.open.ac.uk/ontology/educational.owl#depth_of_coverage	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#comprehensiveness	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#relevancy	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#accuracy	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#educational_level	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#validity	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#rated_by_user	143
http://carre.kmi.open.ac.uk/ontology/educational.owl#categories	127
http://carre.kmi.open.ac.uk/ontology/educational.owl#source	127
http://carre.kmi.open.ac.uk/ontology/educational.owl#alternative_title	110

[17] Medical Educational Resources Aggregator ontology (http://bioportal.bioontology.org/ontologies/MERA) developed in CARRE and described in D.4.2.

Table 27. The SPARQL query to count all educational material linking to CARRE risk elements

```
PREFIX edu: <http://carre.kmi.open.ac.uk/ontology/educational.owl#>
SELECT ?Predicates count(*) as ?Count
WHERE {
   ?s ?Predicates ?o .
    ?o a edu:object
   FILTER regex(str(?Predicates), "^http://carre.kmi.open.ac.uk/ontology/risk.owl")
}
group by ?Predicates
order by DESC(?Count)
```

Table 28. The SPARQL query results: number of educational material linking to CARRE risk elements

Properties of risk ontology	**Count**
http://carre.kmi.open.ac.uk/ontology/risk.owl#has_educational_material	515

Table 29. The SPARQL query to count all triples inserted by Educational Resource Aggregator

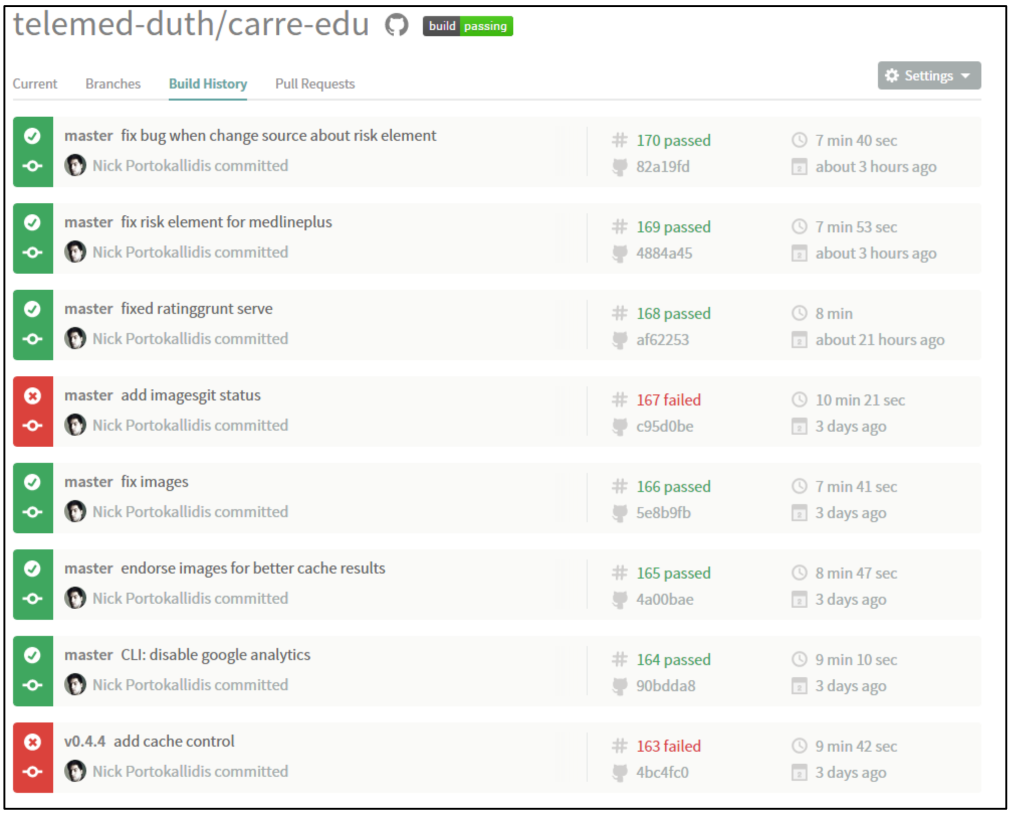
```
SELECT count(?s) as ?All_Educational_Triples
WHERE {
   ?s ?p ?o .
   FILTER regex(str(?s), "^http://carre.kmi.open.ac.uk/public/educational")
}
```

Table 30. The SPARQL query results: number of triples inserted by Educational Resource Aggregator

All Educational Triples
10707

6.2.3. Automated software testing and code optimizations

We have built an automated tests using Jasmine test framework and run them with each new version of the Aggregator. For the test runner we use Travis CI, which is a cloud service. Travis CI is an open-source hosted, distributed continuous integration service used to build and test projects hosted at GitHub. Below we show two screenshots regarding the testing jobs and actual tests that are implemented.

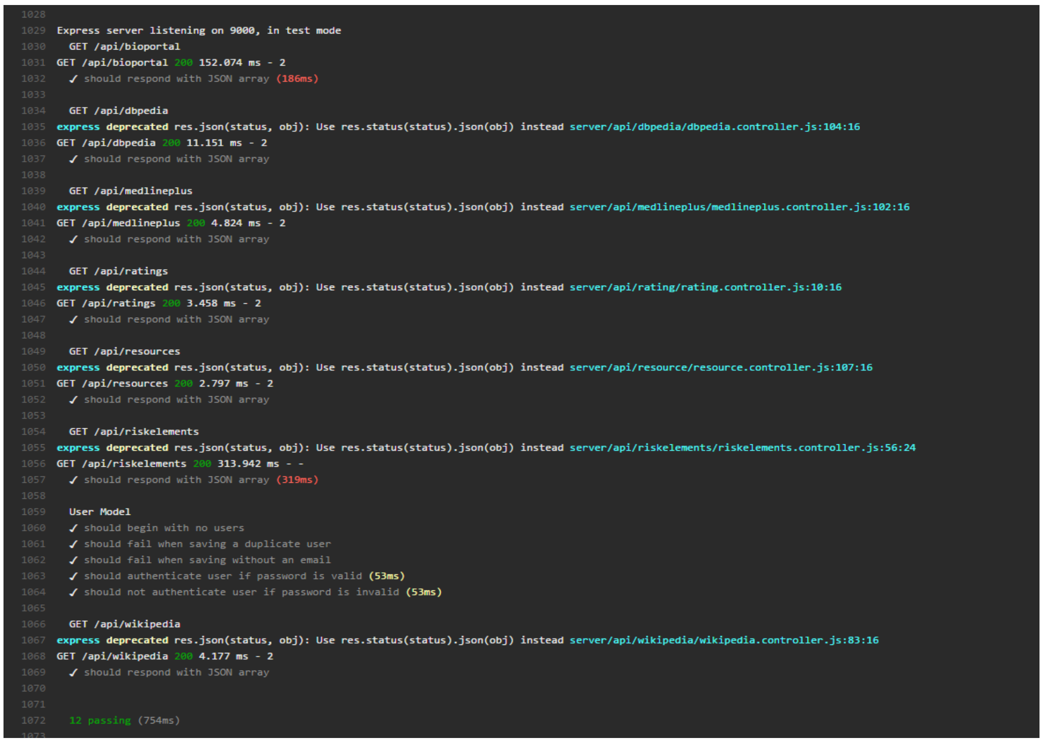


telemed-duth/carre-edu build passing

Current | Branches | **Build History** | Pull Requests | Settings

✓	master fix bug when change source about risk element Nick Portokallidis committed	# 170 passed 82a19fd	7 min 40 sec about 3 hours ago
✓	master fix risk element for medlineplus Nick Portokallidis committed	# 169 passed 4884a45	7 min 53 sec about 3 hours ago
✓	master fixed ratinggrunt serve Nick Portokallidis committed	# 168 passed af62253	8 min about 21 hours ago
✗	master add imagesgit status Nick Portokallidis committed	# 167 failed c95d0be	10 min 21 sec 3 days ago
✓	master fix images Nick Portokallidis committed	# 166 passed 5e8b9fb	7 min 41 sec 3 days ago
✓	master endorse images for better cache results Nick Portokallidis committed	# 165 passed 4a00bae	8 min 47 sec 3 days ago
✓	master CLI: disable google analytics Nick Portokallidis committed	# 164 passed 90bdda8	9 min 10 sec 3 days ago
✗	v0.4.4 add cache control Nick Portokallidis committed	# 163 failed 4bc4fc0	9 min 42 sec 3 days ago

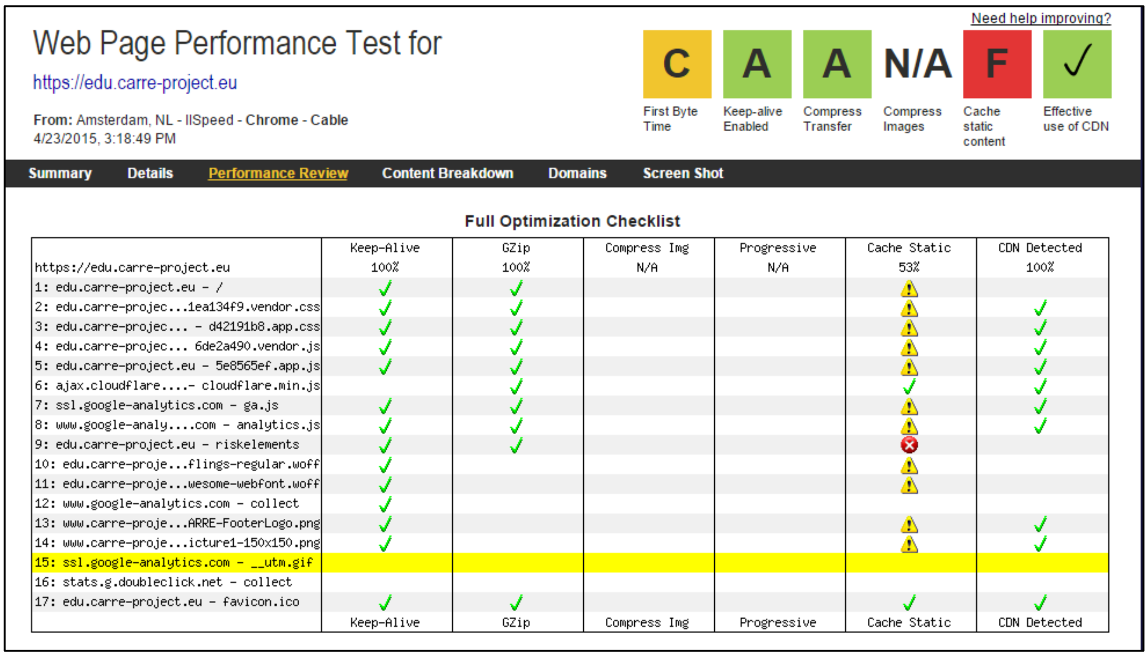
Figure 4. Screenshot of the testing job



```

Express server listening on 9000, in test mode
  GET /api/bioportal
GET /api/bioportal 200 152.074 ms - 2
  ✓ should respond with JSON array (186ms)

  GET /api/dbpedia
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/dbpedia/dbpedia.controller.js:104:16
GET /api/dbpedia 200 11.151 ms - 2
  ✓ should respond with JSON array

  GET /api/medlineplus
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/medlineplus/medlineplus.controller.js:102:16
GET /api/medlineplus 200 4.824 ms - 2
  ✓ should respond with JSON array

  GET /api/ratings
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/rating/rating.controller.js:10:16
GET /api/ratings 200 3.458 ms - 2
  ✓ should respond with JSON array

  GET /api/resources
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/resource/resource.controller.js:107:16
GET /api/resources 200 2.797 ms - 2
  ✓ should respond with JSON array

  GET /api/riskelements
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/riskelements/riskelements.controller.js:56:24
GET /api/riskelements 200 313.942 ms - -
  ✓ should respond with JSON array (319ms)

  User Model
  ✓ should begin with no users
  ✓ should fail when saving a duplicate user
  ✓ should fail when saving without an email
  ✓ should authenticate user if password is valid (53ms)
  ✓ should not authenticate user if password is invalid (53ms)

  GET /api/wikipedia
express deprecated res.json(status, obj): Use res.status(status).json(obj) instead server/api/wikipedia/wikipedia.controller.js:83:16
GET /api/wikipedia 200 4.177 ms - 2
  ✓ should respond with JSON array


  12 passing (754ms)
```

Figure 5. Screenshot of the implemented tests

Along with the automated software testing we also conducted some diagnostic experiments using webpagetest.org, an online web page diagnostic service. Then we tried to fix some of the bugs related to static content caching that were affecting the user experience of the application. Below we show two diagnostic results – before and after the optimization.



Need help improving?

Web Page Performance Test for
https://edu.carre-project.eu

From: Amsterdam, NL - IISpeed - **Chrome** - **Cable**
4/23/2015, 3:18:49 PM

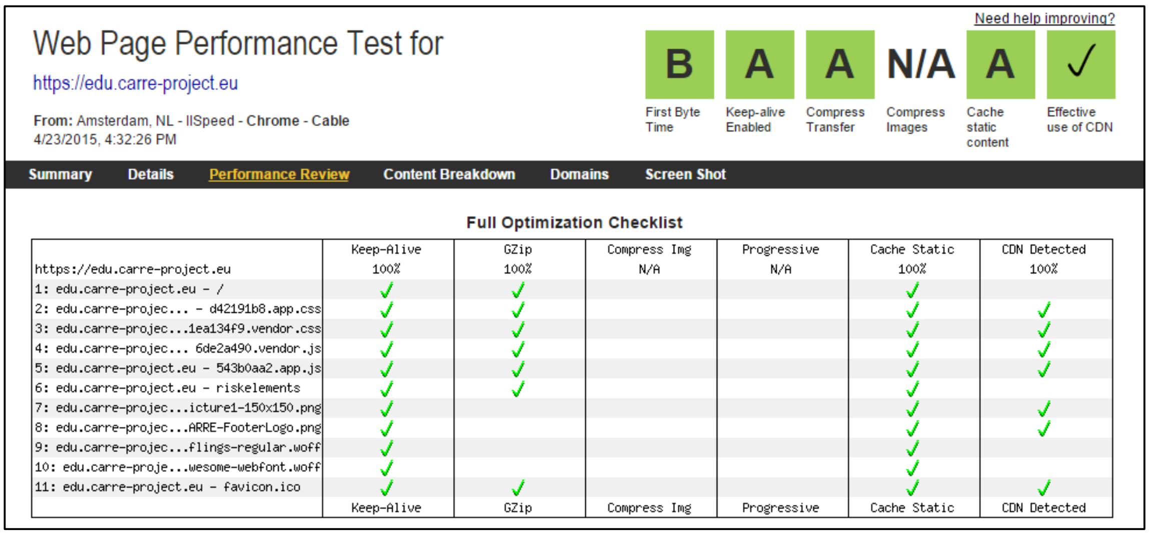
C	A	A	N/A	F	✓
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

Summary | Details | Performance Review | Content Breakdown | Domains | Screen Shot

Full Optimization Checklist

	Keep-Alive	GZip	Compress Img	Progressive	Cache Static	CDN Detected
https://edu.carre-project.eu	100%	100%	N/A	N/A	53%	100%
1: edu.carre-project.eu - /	✓	✓			⚠	
2: edu.carre-projec...1ea134f9.vendor.css	✓	✓			⚠	✓
3: edu.carre-projec... - d42191b8.app.css	✓	✓			⚠	✓
4: edu.carre-projec... 6de2a490.vendor.js	✓	✓			⚠	✓
5: edu.carre-project.eu - 5e8565ef.app.js	✓	✓			⚠	✓
6: ajax.cloudflare....- cloudflare.min.js		✓			✓	✓
7: ssl.google-analytics.com - ga.js	✓	✓			⚠	✓
8: www.google-analy....com - analytics.js	✓	✓			⚠	✓
9: edu.carre-project.eu - riskelements	✓	✓			✗	
10: edu.carre-proje...flings-regular.woff	✓				⚠	
11: edu.carre-proje...wesome-webfont.woff	✓				⚠	
12: www.google-analytics.com - collect	✓					
13: www.carre-proje...ARRE-FooterLogo.png	✓				⚠	✓
14: www.carre-proje...icture1-150x150.png	✓				⚠	✓
15: ssl.google-analytics.com - __utm.gif						
16: stats.g.doubleclick.net - collect						
17: edu.carre-project.eu - favicon.ico	✓	✓			✓	✓
	Keep-Alive	GZip	Compress Img	Progressive	Cache Static	CDN Detected

Figure 6. Diagnostic results before optimization



Need help improving?

Web Page Performance Test for
https://edu.carre-project.eu

From: Amsterdam, NL - IISpeed - **Chrome** - **Cable**
4/23/2015, 4:32:26 PM

B	A	A	N/A	A	✓
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

Summary | Details | Performance Review | Content Breakdown | Domains | Screen Shot

Full Optimization Checklist

	Keep-Alive	GZip	Compress Img	Progressive	Cache Static	CDN Detected
https://edu.carre-project.eu	100%	100%	N/A	N/A	100%	100%
1: edu.carre-project.eu - /	✓	✓			✓	
2: edu.carre-projec... - d42191b8.app.css	✓	✓			✓	✓
3: edu.carre-projec...1ea134f9.vendor.css	✓	✓			✓	✓
4: edu.carre-projec... 6de2a490.vendor.js	✓	✓			✓	✓
5: edu.carre-project.eu - 543b0aa2.app.js	✓	✓			✓	✓
6: edu.carre-project.eu - riskelements	✓	✓			✓	
7: edu.carre-projec...icture1-150x150.png	✓				✓	✓
8: edu.carre-projec...ARRE-FooterLogo.png	✓				✓	✓
9: edu.carre-projec...flings-regular.woff	✓				✓	
10: edu.carre-proje...wesome-webfont.woff	✓				✓	
11: edu.carre-project.eu - favicon.ico	✓	✓			✓	✓
	Keep-Alive	GZip	Compress Img	Progressive	Cache Static	CDN Detected

Figure 7. Diagnostic results after optimization

6.2.4. Summary of testing results

In relation to the test scenarios and cases that were described in Table 19 and Table 20, we present only the fixed cases along with some details of each scenario in the next table. The following test cases involve only the functional and non-functional testing of the Educational Resource Aggregator.

Table 31. Discussion and solutions about the test cases with issues.

TS_F_ED_05.TC02	Details: Metadata Enrichment and Mapping CARRE schema Results: Resources linked to CARRE risk elements Fixed.
TS_F_ED_08.TC01	Details: Login/Logout and Register through CARRE devices through https

TS_F_ED_08.TC02	Results: Fixed by using online CDN and SSL services for secure proxy
TS_F_ED_10.TC03	Details: Navigate to educational resource Results: Resource showing in Iframe compatibility with source website Fixed
TS_N_ED_01.TC03	Details: Security Results: Transmit data through encrypted protocol (https). A new SSL certificate generated Fixed
TS_N_ED_02.TC04	Details: Data security testing Results: User authentication saved temporarily in encrypted browser session Fixed
TS_N_ED_06.TC01	Details: Scalability Testing Results: Fixed by removing local data store unit dependency

7. Testing of RESTful API

7.1. Specific testing methodology and tools

The *RESTful API* component exposes a number of CARRE-related services that allow accessing RDF data, both public and private. A complete presentation of the RESTful API component can be found in D.4.1. This section reports on the performance testing of the RESTful API.

Currently, the RESTFul API implements a number of methods that expose private and public RDF data. The output of these methods are producing responses in JSON format, which allow developers within CARRE, as well as third-party interested stakeholders, to build web services and applications on top of them. A crucial parameter in the success of this component is its robustness and high performance.

A performance aspect that was taken into account during the development of the component concerns the access of RDF data that reside in remote repositories. To achieve this, our component implements federated queries[18]. In order to overcome high response times from these external data sources, CARRE's RESTFul component implements a caching mechanism that allows the fast execution of such methods.

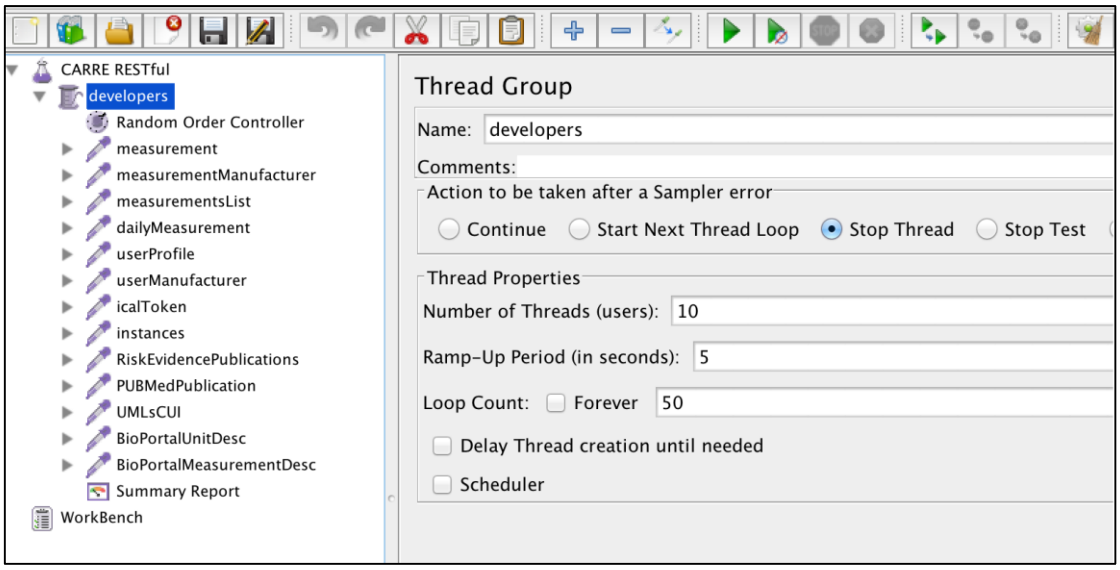
For the assessment of the component's performance, we have used Apache JMeter[19], which is a Java application designed to load test functional behaviour and measure performance. More specifically, the settings of this testing are the following:

- 10 threads, corresponding an equivalent number of users/applications are deployed every 5 seconds. This simulates the maximum load expected for our repository.
- Each one of the methods is tested randomly. A testing *cycle* is complete once all methods have been accessed.
- In total, 50 *cycles* are executed continuously. In total, the methods are accessed 500 times.

Figure 8 shows a screenshot of the settings discussed above.

[18] http://www.w3.org/TR/sparql11-federated-query/

[19] http://jmeter.apache.org/



CARRE RESTful
- developers
  - Random Order Controller
  - measurement
  - measurementManufacturer
  - measurementsList
  - dailyMeasurement
  - userProfile
  - userManufacturer
  - icalToken
  - instances
  - RiskEvidencePublications
  - PUBMedPublication
  - UMLsCUI
  - BioPortalUnitDesc
  - BioPortalMeasurementDesc
  - Summary Report

WorkBench

Thread Group

Name: developers

Comments:

Action to be taken after a Sampler error

Continue [✓ Stop Thread] Start Next Thread Loop Stop Test

Thread Properties

Number of Threads (users): 10

Ramp-Up Period (in seconds): 5

Loop Count: ☐ Forever 50

☐ Delay Thread creation until needed

☐ Scheduler

Figure 8. A screenshot of Apache JMeter configuration.

7.2. Summary report of CARRE's RESTful API performance testing

Table 32 shows the summary report from the Apache JMeter performance testing. The table shows that no errors produced during the testing. The first rows of the table report on methods concerning fetching of measurement data. These methods have a response time of a few seconds and produce relatively big JSON objects. Other methods such as user profile lookup are very fast and are produced in a few milliseconds (ms). Furthermore, methods that invoke external data sources appear to respond very fast. This low response time is due to the caching mechanism implemented (see discussion in previous section). Finally, the most intense method constitutes the Calendar-based generation of summary report for a user. This method is typically accessed once or twice daily by a calendar application and is indeed expected to be computationally expensive, since it scans all triples within a private graph and computes aggregates for each measurement and day.

Table 32: Apache JMeter summary report for CARRE's RESTFul API.

Method	# Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev. (ms)	Error %	Throughput (requests/min)	KB/sec	Avg. Bytes
measurement	500	2783	1044	9630	463.99	0.00%	28.30	5.53	11997
measurementManufacturer	500	701	443	7504	353.66	0.00%	28.34	125.8	272722
measurementsList	500	1331	530	7917	364.25	0.00%	28.34	0.55	1188
dailyMeasurement	500	1465	528	1919	208.2	0.00%	28.34	0.12	262
userProfile	500	115	58	6361	285.7	0.00%	28.38	0.12	256
userManufacturer	500	174	101	6654	410.83	0.00%	28.38	0.12	252
icalToken	500	12615	4846	19529	1135.57	0.00%	28.26	26.03	56606
instances	500	317	116	977	138.14	0.00%	28.62	39.19	84117
RiskEvidencePublications	500	98	55	613	53.75	0.00%	28.63	0.08	170
PUBMedPublication	500	544	59	1296	315.65	0.00%	28.62	2.74	5892
UMLsCUI	500	642	74	1293	322.77	0.00%	28.62	7.85	16863
BioPortalUnitDesc	500	164	53	1621	99.08	0.00%	28.64	0.07	152
BioPortalMeasurementDesc	500	251	52	1143	112.09	0.00%	28.64	0.07	152
TOTAL	6500	1631	52	19529	3281.35	0.00%	364.70	205.76	34663.8

Annex 1 Beta Testers Consent Form

Consent form of BETA Testers
in CARRE Project

I, the undersigned ____ (first name, last name), born on the _______________ (date), in ______________________________________ (city, country) and resident at ____ (address), reachable via __________________________________ (e-mail-address), declare by the present consent form my agreement to the processing of my personal data (medical or not) on the CARRE private RDF repository in particular by transferring data from (please check that you agree):

- ☐ Sensor Data Aggregator
  - ☐ Fitbit
  - ☐ Medisana
  - ☐ Withings
  - ☐ iHealth
  - ☐ eMotion Faros
  - ☐ Google Fit
  - ☐ Moves
- ☐ Personal Health Records Aggregator
  - ☐ Vivaport
  - ☐ HealthVault
  - ☐ Manual Data Entry System
- ☐ Web Lifestyle Data Aggregator
  - ☐ Only intentions from web searches (Google, Bing and Yahoo) that are relevant to CARRE (travel and health intentions) and not the actual web queries.

for the purposes of scientific development and validation of the European research project CARRE (Grant agreement no: 611140) (http://carre-project.eu/).

The CARRE project investigates information and communication technologies for empowering patients with comorbidities (multiple co-occurring medical conditions), or persons with increased risk of such conditions, especially in the case of chronic cardiac and renal disease patients. At the present time, as part of the technical development of this environment, the CARRE project wishes to use the data for the testing of implemented aggregators that was developed in this context. The intention is to allow the data collected by the above mentioned apps, devices and services to be linked within the CARRE private RDF repository only, and accessible to each user in low level.

I am aware that all necessary state-of-the-art security measures are incorporated in the platform to protect my data against accidental or unlawful destruction or accidental loss, alteration, unauthorized disclosure or access or any other misuse.

I understand and agree that all data that I collect and provide to the project by using the above mentioned services may for the project duration be stored and used by the institutions participating in the project (as listed on the project website) in a public cloud that may use servers located outside the EU/EEA (and may provide a level of privacy protection lower than that offered by EU data protection legislation).

Furthermore, I am aware of the fact that the above mentioned devices, apps and the services are subject to their own third party privacy rules (from the device-manufacturers, e.g. FitBit) and that the project has no control over data processing by such parties.

I understand that the institutions participating in the project are the only entities which have access to the data which I have uploaded to the private RDF repository of every user.

In case of any change to the above position, and in particular if the functions of CARRE will change, if additional apps, devices and other services will be linked to the CARRE private RDF repository, or if it is planned to make the demo platform public or use the data for any other purposes than those mentioned, the project will inform me by using my address or e-mail-address (as specified by me) for additional consent.

I have been given the opportunity to ask questions about the processing of my data and I have had these answered satisfactorily.

I am aware that my participation is voluntary and that I will not suffer adverse consequences for refusing to grant consent. I understand that I have the right at any time to withdraw my consent to the processing of my data on the platform without giving any reason. In the event of wishing to do so or having other concerns I may contact the coordinator Prof. Eleni Kaldoudi (kaldoudi@med.duth.gr) at Democritus University of Thrace. In this case, my uploaded data will be permanently deleted from the private RDF repository.

A copy of this agreement will be sent to my address/e-mail-address (as specified by me) and another copy will be retained for record-keeping by the project.

____

DD MM YYYY, “First name, last name, signature”