MICS

Measuring Impact of Citizen Science

Developing metrics and instruments to evaluate citizen science impacts on the environment and society

EC Horizon-2020 Grant Agreement number 824711

Call: H2020-SwafS-2018-2020 (Science with and for Society)

Topic: SwafS-15-2018-2019

Type of action: RIA

Deliverable D.3.1: Report on the technical requirements

Delivery year: 2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824711.

Document Information

Project Number	824711	**Acronym**	MICS
Full title	Developing metrics and instruments to evaluate citizen science impacts on the environment and society		
Project URL	**www.mics.tools**		
EU Project officer	Colombe Warin		

Deliverable	**Number**	D3.1	**Title**	Report on the technical requirements
Work package	**Number**	3	**Title**	Toolboxes for methods application, information visualisation, and delivery to decision makers, citizens and researchers

Date of delivery	**Contractual**	Month 05 (May 2019)	**Actual**	Month 06 (June 2019)
Dissemination Level	Public			

Authors (Partner)	Earthwatch		
Responsible Author	Patino Velasquez, L.F.	**Email**	lfvelasquez@earthwatch.org.uk
	Partner	Earthwatch	

Abstract (for dissemination)	This document sets general, best-practice principles for developing the MICS platform, analyses tools and projects' results that can be adapted and adopted for developing the MICS platform, and contains initial functional requirements for the development of the MICS platform.
Keywords	Toolbox, development

Version Log			
Version as date	**Author**	**Partner**	**Change or comments** (optional)
2019_02_12	Patino Velasquez, L.F.	Earthwatch	Initial document creation
2019_03_15	Patino Velasquez, L.F.	Earthwatch	System characteristics
2019_05_10	Dan Simota	GeoEcoMar	System architecture overview

2019_05_13	Patino Velasquez, L.F.	Earthwatch	System overview and system architecture
2019_05_15	Patino Velasquez, L.F.	Earthwatch	First draft of the complete document
2019_05_21	Patino Velasquez, L.F.	Earthwatch	Document re-structuring following feedback
2019_06_03	Nigel Hussain	Earthwatch	State of the art
2019_06_07	Patino Velasquez, L.F.	Earthwatch	MICS platform development criteria and principles
2019_06_12	Patino Velasquez, L.F.	Earthwatch	Document preparation for review by consortium
2019_06_25	Uta Wehn	IHE Delft	Review
2019_06_27	Luigi Ceccaroni	Earthwatch	Final review

To cite this document:

Patino Velasquez, L.F., Hussain, N., Simota, D. and Ceccaroni, L. (2019). D3.1: Report on the technical requirements. *Deliverable report of project H2020 MICS (grant agreement No 824711)*.

Table of contents

1. Introduction

The MICS project develops approaches and tools to evaluate citizen-science impacts.

The MICS project specifically aims to, among other things (The full list of objectives can be found in the DoA and at [https://www.mics.tools/about-mics].):

- provide comprehensive, participatory and inclusive **metrics and instruments** to evaluate citizen-science impacts;
- implement an impact-assessment knowledge-base through toolboxes for methods application, information visualisation, and delivery to decision makers, citizens and researchers.

The result of achieving these objectives is an integrated **platform** where these metrics and instruments are available for use by anyone involved in a citizen-science project wanting to understand its impact, whether at the planning stage or several years after the project's conclusion.

1.1. Purpose

This document contains initial functional requirements for the development of the MICS platform. The information in this document aims to attain the following objectives:

- to analyse state-of-the-art technology in relation to web-application tools and features;
- to outline a set of principles and guidance for the development of an operational prototype of the MICS platform;
- to present and define an authentication system for the MICS platform;
- to present a solution for technical performance monitoring;

1.2. Scope

This document provides a set of best practices and principles recommended for the development of the MICS platform. It discusses the characteristics and technical requirements for the development of the platform and its translation into a web application.

A high-level view of the state of the art concerning architecture, frameworks, monitoring and authentication is presented together with publicly available assessment-tools developed as web application.

The specification within this document will be implemented in the development of an operational prototype and ultimately the final system. In order to coordinate efforts between the different partners and encourage transparency, this document includes a description of standards, formats and technical conventions.

1.3. Acronyms and abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
APM	Application Performance Monitoring

COST	European Cooperation in Science & Technology
DoA	Description of the Action
DRY	Don’t Repeat Yourself
ECSA	European Citizen Science Association
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LTS	Long Term Support
MVC	Model – View – Controller
NBS	Nature-Based Solution
OSS	Open Source Software
REST	Representational State Transfer
SaaS	Software as a Service
SPA	Single Page Application
UTF	Unicode Transformation Format
WP	Work Package
XML	Extensible Mark-up Language
XSS	Cross-site scripting

1.4. Overview

This document is structured in two main sections as follows:

Section two presents the state of the art concerning web-application development. In here we describe concepts with regards to application’s architecture, frameworks, authentication and authorization, and web-application monitoring and performance. In addition, this section provides information concerning existing online assessment-tools in fields such as health and economy, amongst others.

Section three describes the future development of the MICS platform by presenting a set of web-application design principles, as well as framework selection criteria and best practices that will be followed throughout the implementation of the operational prototype and final solution of the MICS platform. In addition, this section discusses the characteristics of authentication and application monitoring system to be considered in the development. Finally, the section presents the draft version of the user’s interface wireframes developed. The final part of this section describes standards of importance for the development of the MICS platform.

2. State of the art

This section presents the technological state of the art that will be considered during the development of the MICS platform.

2.1. Web-application architecture

Technological advances in the area of web applications have given place to complex and sophisticated software solutions (Madeyski & Sochmialek, 2005; Gordillo et al., 2006).

Before starting a web-application development project, it is important to choose both the type of web-application architecture and the model of the web-application components as the architecture and model will guide the next steps in development.

The web application architecture describes the interactions between applications, databases, and middleware systems on the web. Web application architecture has to not only deal with efficiency, but also with reliability, scalability, security, and robustness.

The application's architecture could be described as the structure of a system focused on arranging the different components that enable and support specific functionalities. Importantly, this arrangement of components is commonly referred to as "areas of concern" (Meier et al., 2008). Figure 1 illustrates a general application-architecture grouping the different components by areas of concern.

2.1.1 Types of web-application architecture

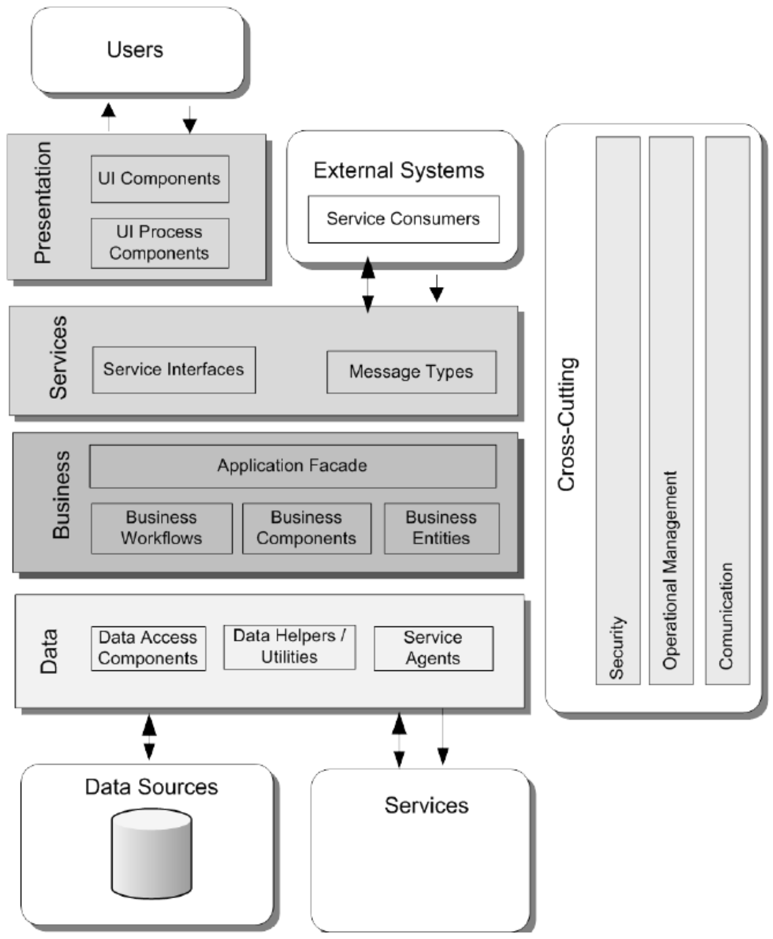
The type of web application architecture depends on how the application logic is distributed among the client and server sides. There are three primary types of web application architecture:

Single-page applications **(SPAs):** instead of loading completely new pages from the server each time for a user action, SPAs allow for a dynamic interaction by means of providing updated content to the current page. AJAX, a concise form of Asynchronous JavaScript and XML, is the foundation for enabling page communications and, hence, making SPAs a reality. Because single-page applications prevent interruptions in user experience, they, in a way, resemble traditional desktop applications. However, there are a few downsides to using SPAs:

- **dependencies:** SPAs are built with JavaScript, so JavaScript should be enabled in client browser.
- **security**: Compared to a traditional page, a SPA is less secure due to *cross-site scripting* (XSS).
- **memory leak**: A memory leak in JavaScript can cause even the most powerful system to slow down.

SPAs are designed in a way so that they request for most necessary content and information elements. This leads to the procurement of an intuitive as well as interactive user experience.

Figure 1. Common web-application architecture (Meier et al., 2008)



Microservices: these are small, and lightweight services that execute a single functionality, connected by API's. The microservices architecture framework has a number of advantages that allows developers to not only enhance productivity but also speed up the entire deployment process.

The components making up an application build using the microservices architecture aren't directly dependent on each other. As such, they don't need to be built using the same programming language. Hence, developers working with the microservices architecture are free to pick up a technology stack of choice. This practice streamlines development by making it simpler and quicker.

However, there are a few disadvantages documented by the web application development community to using the microservices framework [https://www.qat.com/15-benefits-microservices]. These are outlined below:

- **testing difficulties**: Automated testing becomes more challenging when each microservices is running on different runtime environment;
- **decreased performance**: In the microservice architecture, each service runs as an independent process (multiple java virtual machine (JVM) instances). However, in a

monolithic application, all services are part of single shared process (single JVM instance). As a shared process communication is faster than an inter-process communication, the microservice's performance is slightly worse than that of a monolithic application;

- **increased memory consumption**: Applications built with a microservice architecture replaces N monolithic application instances with N (monolithic) instances x M (microservices) instances. If each service runs in its own JVM (or equivalent), which is usually necessary to isolate the instances, then there is the overhead of M times as many JVM runtimes. In addition, when services are deployed on multiple machines, there might be several utility classes and libraries that will get replicated. Consequentially, the application will have an overall higher memory footprint
- **deployment complexity**: If services span multiple systems then operational cost of deploying and managing those services and systems will be greater due to the many deployment configuration files and job scripts running;
- **coding complexity**: Developers must properly implement the inter-service communication mechanism. Implementing use cases that span multiple services without using distributed transactions will need extra efforts. Implementing use cases that span multiple services would requires careful coordination between the teams;

Serverless architectures: in this type of web-application architecture, an application developer consults a third-party cloud infrastructure services provider for outsourcing server as well as infrastructure management. The benefit of this approach is that it allows applications to execute the code logic without interfering with the infrastructure-related tasks. The serverless architecture is preferable when the development company does not want to manage or support the servers as well as the hardware they have developed the web application for. Once again, some of the disadvantages include [https://www.techpally.com/serverless-architecture-disadvantages/]:

- **vendor exclusivity**: The architecture is dictated by the vendor. As the servers are in the hands of third-party providers, we do not have control over the run times, updates and even the hardware. This leads to inconsistencies and a limitation of resources. In addition, if we switch vendors, we would have to invest even more time, effort and resources into reengineering the software. Finally, the vendor can revise its service terms or pricing policies as and when required, and even cease offering the option;
- **long-term tasks**: It is found that a serverless architecture is one of the best options for a short-term process. But when the duration of your task increases, there would be functionalities that might need to be executed more often, leading to increased payments for the time these functionalities run and making serverless architecture ill-suited for long-term tasks;
- **complexity**: It is not easing to get a hold of how serverless applications operate. It might take a lot of effort. The integration units that come in serverless are smaller than those you might find in other architectures. This might need you to take some time to organize the functions such that they work in sync with the data. You might have problems with the versioning and deployment side.

2.2. Web-application frameworks

Advances in Web 2.0 and online technologies have transformed the development of web-based software by making it as influential and significant as desktop-based solutions. Developing web applications able to provide high levels of functionality is now a complex task requiring evolving toolsets and in the majority of cases multiple developers, and this level of complexity has resulted in the creation and implementation of frameworks for web application development.

Frameworks are a high-level development environment in which programmers are able to reuse software pieces making the process of developing applications much easier and faster. Frameworks comprise source code libraries, and a wide range of tools amongst other characteristics that accelerate the development pace ensuring a better level of quality of the final product (Chao et al., 2013).

The use of frameworks for web application development is not a paramount process. However, a framework provides the development team/individual with the certainty that the application in development is in full compliance with the requirements, that is structured, well-tested, and that is both maintainable and upgradable. More importantly, any framework independent of the technology used for their development ought to have the following minimum set of characteristics (Hu et al., 2008):

- based on the mature web framework, the software developer does not need to directly contact the bottom of the API, just write some necessary code. It simplifies the developed process, and then improves system stability and operational efficiency;
- each mature web framework has a professional team to provide full-time work by offering the frame for free to reduce development cost;
- simplifying development model to easily separate the user interface and navigation from the business logic;
- the distinct system structure can be provided by well-designed web framework, increasing the cohesion of the system. Good structure makes it easier for other people to join the project;
- an easy-to-use web framework offers some of examples and documentation for users to optimal practice;
- the code of a mature web framework often has been tested in various application environments, simplifying the software developers' code testing process.

The two main reasons for choosing a framework are described in the following two subsections.

2.2.1. Guaranteed upgradability and maintenance

The structure that a framework provides for the application allows any developer to easily "adopt" an application, maintain it over time, and to upgrade it both quickly and neatly, whenever necessary. Thus, in the long run, a framework ensures the longevity of your applications.

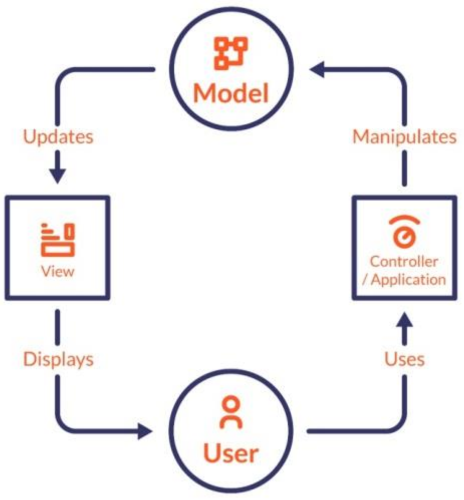
2.3. Framework architecture

A framework is a high-level solution for the reuse of software pieces. The framework delivers application behaviour at a high level of abstraction by providing functionalities within a distinct domain, defining interaction patterns between known components, and providing enough flexibility to be tailored to tangible context (Chao et al., 2013).

Further, these provide reuse of code and design. An example of the framework is the Model-View-Controller (see Figure 2. Model - View – Controller architecture). The architecture itself is specifically designed to separate the internal representations of the information from the ways the information is presented to the user. As a result, the framework is sufficient at separating business logic from presentation logic. To accomplish this, the MVC framework separates the project into three components[1]. They are:

- **the model**: This component handles the application logic for the application data and it usually interacts with the file, database, and the web service. It is here where data passing and leaving the controller and in some particular cases the view is manipulated, this process takes place without having a direct link to HTTP and/or web servers;
- **the view:** component is also known as presentation layer, thus encompassing the functionalities responsible for the interaction of the end user with the system, and, in most cases, this corresponds to some form of UI;
- **the controller**: This component as per its name controls the application logic and flow of the application execution. This is achieved by passing inputs through the view component and directing it to the modal component for processing and logical operations, and gathering the end product of the request and sending it back using the view component. MVC application should have one or more controller as each controller contains a class that represents a different method (Reenskaug and Coplien, 2013).

Figure 2. Model - View – Controller architecture



[1]*Burbeck (1992), p.2: "the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object."*

2.4. Authentication and authorization system

Proliferation of web applications and in particular the use of a centralised and interoperable login system has become imperative. Social login, the ability to sign-in to a website using online identity from provider like Facebook, Google and Twitter, continues to dramatically increase amongst web application end user, as using an existent identity in order to bypass the traditional online registration process has become a more common practice (Gafni and Nissim, 2014).

Social login only concern is establishing the identity of the user and then sharing that information with each subsystem that requires the data.

There are multiple solutions for implementing social login, the two most common web security protocols outside enterprise deployments are OAuth2, OpenID Connect (Naik et al., 2017).

2.4.1. OAuth2

OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 supersedes the work done on the original OAuth protocol created in 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification is being developed within the IETF OAuth WG.

OAuth2 provides secure delegated access, meaning that an application, called a client, can take actions or access resources on a resource server on the behalf of a user, without the user sharing their credentials with the application. OAuth2 does this by allowing tokens to be issued by an identity provider to these third-party applications, with the approval of the user. The client then uses the token to access the resource server on behalf of the user [https://oauth.net/2/].

2.4.2. OpenID Connect

OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol. OpenID Connect is an authentication layer on top of OAuth 2.0, an authorization framework. The OpenID Foundation controls the standard. It allows Clients to verify the identity of the end user based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the end user in an interoperable and REST-like manner.

OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them. [http://openid.net/connect/]

2.5. Monitoring and ensuring system performance

Once a platform has been deployed, the technical performance should be monitored to ensure that all functionalities are running adequately and all the components are working properly, as well as, to gain a greater understanding of the platform use.

Whilst monitoring and ensuring web application performance there are two types of metrics that are commonly monitored:

- **concurrency and throughput:** this is the largest number of concurrent system users that the system is expected to support at any given moment. For example, if the web application has the maximum performance of the server, this will be able to have the highest number of concurrent system users;
- **server response time:** this refers to the time taken for one system node to respond to the request of another system node. A simple example would be a HTTP 'GET' request from browser client to web server. In terms of response time this is what all load testing tools actually measure. It may be relevant to set server response time goals between all nodes of the system. For instance, if a web application has higher server response time, then this will have a better user experience.

The benefits of this kind of testing include:

- improving user experience;
- carrying out important metrics to fine-tune the systems;
- identifying bottlenecks in the database configuration;
- comparing actual performance compared with expectations.

In addition, there are several kinds of test to measure system performance, the most common test within the web development community are:

- **load testing**: this test will give out the response times of all the important business critical transactions. The database, application server, etc. are also monitored during the test; this will assist in identifying bottlenecks in the application software and the hardware that the software is installed;
- **stress testing**: this kind of test is done to determine the system's robustness in terms of extreme load. It also helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum;
- **soak testing**: is usually done to define if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect any potential leaks. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

2.6. Assessment tools

This section presents the state of the art in assessment tools that have been designed as online applications. These assessment tools have implemented some of the characteristics described in previous sections of this document; additionally, by exploring these tools, we had the opportunity to identify features necessary for a successful implementation of the concepts and methodologies that form part of a platform.

Table 1. Assessment tools description" presents a description of the assessment tools. They were selected taking into consideration characteristics inherent to the MICS project, such as online (free) availability, user interactivity, use of quality assurance process, and consideration of many types of

activities. In addition to these characteristics, the selection also considered the usability web principles defined by Abrahão et al. (2008):

- web application learnability: ease of understanding the content and services made available through the assessment tool;
- web application efficiency: content accessibility through the links available within the assessment tool;
- memorability: orientation around the assessment tool after a period of inactivity;
- few errors;
- user satisfaction.

Table 1. Assessment tools description

Assessment tool	Description
PIA software	This software tool facilitates carrying out a data protection impact assessment.
The SDG impact assessment tool	A self-assessment tool for learning the impact of an activity, organisation or innovation over the SDGs
IMPACT	Tool developed by Aquatera to identify potential impacts of marine energy development on Scotland's marine ecological environment
***Health economic assessment tool* (HEAT) for walking and cycling**	The HEAT tool is designed to enable users to conduct economic assessments of the health impacts of walking or cycling.
Ramboll's SDG assessment tool	This tool carries out a high-level assessment of SDG impact or potential.
The B impact assessment tool	Tool designed to measure the impact of companies on community, environment and customers

2.6.1 PIA software

This software tool facilitates carrying out a data *protection impact assessment* (PIA). The software, guidelines, and assessment tools can all be found at [https://www.cnil.fr/en/open-source-pia-software-helps-carry-out-data-protection-impact-assesment]. As the MICS tools will involve much data flow and will potentially involve sensitive pieces of information including names and emails, having an important data protection is vital to ensure the success and continued funding of the project.

The PIA tool has been designed around three principles:

- **A didactic interface to carry out PIAs**: the tool relies on a user-friendly interface to allow for a simple management of PIAs. It clearly unfolds the privacy impact assessment methodology step by step. Several visualisation tools offer ways to quickly understand the risks.

- **A legal and technical knowledge base:** the tool includes the legal points ensuring the lawfulness of processing and the rights of the data subjects. It also has a contextual knowledge base, available along all the steps of the PIA, adapting the contents displayed.

- **A modular tool:** designed to help build compliance, the tool contents can be customised to specific needs or business sector. Published under a free licence, it is possible to modify the source code of the tool in order to add features or include it into tools used in your organisation (see also [https://www.fsb.org.uk/resources/why-is-data-protection-so-important]).

As a result of these design principles, using the PIA software will ensure that the MICS project will adhere to the principles laid out by the EU's GDPR, according to which data are:

- only used in specifically stated ways;
- not stored for longer than necessary;
- used only in relevant ways;
- kept safe and secure;
- used only within the confines of the law;
- not transferred out of the European Economic Area;
- stored following people's data protection rights.

2.6.2 Impact assessment tools

Impact assessment tools exist for learning the impact of an activity, organisation or innovation over the *sustainable development goals* (SDGs). In particular, the SDG impact assessment tool [https://sdgimpactassessmenttool.org/] and Ramboll's SDG assessment tool [**Error! Hyperlink reference not valid.**] are free online learning tools that visualise the results from a self-assessment of how an activity, organisation or innovation affects the SDGs. These aim to stimulate the user to get a better understanding of the complexity of sustainable development and the different aspects of the SDGs.

Although the SDGs should be implemented by nations, they also represent a framework towards which any activity can be evaluated.

To work with sustainable development and the implementation of the SDGs can be hard since it entails almost all aspects of human societies. Furthermore, our knowledge of human societies, the environment and the Earth systems continuously grow. Any SDG impact assessment is dependent of the knowledge level and ambition of the person performing the assessment. Hence, it is inherently subjective and preliminary, and should be open for revision [https://sdgimpactassessmenttool.org/about].

In MICS, the reason for exploring these tools is that they promote trans-disciplinary thinking and a deeper understanding of the SDGs and how they can be related to the project. This allows to reflect on the project's results and has the potential of unlocking new, sustainable actions to drive long-lasting transformation in citizen science.

2.6.3 Additional projects to be considered

The **IMPACT tool** was developed by Aquatera to identify potential impacts of marine energy development on Scotland's marine ecological environment and can be found at [http://www.marine-impact.co.uk/assessment-tool.asp?cat=2]. This is a single page application with a dynamic list of questions, which change depending on the inputs to the previous questions. This model provides a possible blueprint to build the MICS web application.

The **HEAT tool** is designed to enable users to conduct economic assessments of the health impacts of walking or cycling. The format and design of this tool are of particular interest to MICS, as it provides a blueprint of what the MICS toolbox interface could become. The HEAT tool asks the participant what they want to assess, the user inputs the appropriate data, the tool then performs various calculations to determine the impacts of the activity. The flexibility of the approach given the questions is of particular interest, and could be adopted by MICS. More information can be found at [https://www.heatwalkingcycling.org/#how_heat_works].

The **B impact assessment tool** is designed to measure the impact of companies on community, environment and customers. In particular, what is most interesting for MICS are the visualisation and the metrics used. The assessment tool uses a combination of charts and graphs to help companies visualise how they compare to other companies as well as reports with detailed statistics. This will inspire MICS on how to display the results of the analysis. However, MICS will try to streamline the questions asked to ensure a straightforward approach. More information on the tool can be found at [https://bimpactassessment.net/].

3. MICS Platform

This section presents the guidelines, principles and practices that will be followed throughout the development of the MICS platform.

3.1. MICS web-application

Advances in web technology jointly with the evolution of WEB 2.0 have provided the vehicle through which building and developing web applications have benefited and improved (Musser and O'Reilly, 2006). The adoption of web-based software presents the possibility of creating complex web applications able to digest and present real-time data, perform transaction and increase the level of user interactive experience, thus making this technology as powerful and important as desktop software (Plekhanova, 2009).

Some of the advantages of developing the MICS platform using web technologies have been described by Dogan et al. (2015) and these include:

- automatic upgrade with new feature for all users;
- universal access from any machine connected to the Internet;
- being independent of the operating system of users.

During the evaluation of the type of application (see section "2.1.1 Types of web-application architecture" 0) and the web technology to be utilised for the development of the MICS platform, the development team will review and follow, where applicable, the guidelines / design considerations proposed by Meier et al. (2008) outlined below:

- separate the areas of concern;

- a component or an object should not rely on internal details of other components or objects;
- do not duplicate functionality within an application;
- identify the kinds of components you will need in your application;
- group different types of components into logical layers;
- you should not mix different types of components in the same logical layer;
- do not overload the functionality of a component;
- understand how components will communicate with each other;
- keep the data format consistent within a layer or component;
- keep cross-cutting code abstracted from the application business logic as much as possible;
- be consistent in the naming conventions used.

3.2. Web-application framework

Web-application frameworks bring benefits such as open source software solutions, community support and robust documentation (Prokofyeva and Boltunova, 2017).

With a myriad of frameworks for developing web application, the development team of the MICS project will select a framework based on:

- the criteria for comparing frameworks defined and used regularly by the web development community (e.g., Symphony [https://symfony.com/ten-criteria], [https://hackernoon.com/how-to-choose-a-framework-ea8b5b1e1f44], IDEAS2IT [https://www.ideas2it.com/blogs/right-php-framework/]) (see also Table 2. Criteria for comparing frameworks web development community);");
- the set of best practices defined by del Pilar Salas-Zárate et al. (2015) (see also Table 3. Best practices for frameworks selection.").

Table 2. Criteria for comparing frameworks web development community

Category	Factor	What it is
Ecosystem	History and longevity	How mature is the framework? Why was it created?
	Popularity of framework	How widely used is the framework?
	Corporate support	Is there a corporate entity involved as a sponsor or interested party?
	Community and ecosystem	Is the framework supported by a large community? Is there a healthy ecosystem of plugins and libraries that extend core functionality?
Framework	Getting started experience and learning curve	How quickly can a new developer start to use the framework? How hard is it to

		use as applications get more complex?
	Skills required	What skills does a developer need to have in order to be productive with this framework? Do they need to learn syntax or patterns that are specific to the framework itself?
	Completeness of offering	Does the framework provide everything "in the box" or do developers need to provide their own solutions to solve common problems?
	Performance factors	How does this framework perform in a complex application? What approaches does it take to help me make my apps run faster?
	Beyond the browser options	Can this framework be used in authoring non-browser apps, like mobile and desktop?
Tooling	UI & component libraries	Are there UI & component libraries available for this framework?
	IDE & tooling support	Is there support for this framework in my IDE or other popular IDEs?
	Companion & CLI tools	What kind of tooling is available to help me create and manage apps with this framework?
Enterprise	Licensing	Under what license is this framework maintained? Does this license conflict with my enterprise's use of the tool?
	Support & upgrade paths	Do the maintainers of this library provide long-term support (LTS) versions? Are there enterprise support options available?
	Security	How do the maintainers handle security issues? How are security patches distributed?
	Talent pool & resources	How easy is it to hire developers who already know

		this framework, or who can learn it easily?

From Satrom, B., Choosing the Right JavaScript Framework for your Next Web Application. Available at [https://softarchitect.files.wordpress.com/2018/03/choose-the-right-javascript-framework-for-your-next-web-application_whitepaper1.pdf] (accessed 17 May 2019).

Table 3. Best practices for frameworks selection (del Pilar Salas-Zárate et al., 2015)

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https:// cakephp.org]	Lift	Grails	Django	Catalyst
AJAX support	Yes	Yes (Prototype, script.aculo. us, jQuery, among others)	Yes	Yes (Prototype, script.aculo.us, jQuery, MooTools [https:// mootools. net/], among others)	Yes	Yes (jQuery, prototype, Dojo, YUI, MooTools, among others)	Yes (jQuery, prototype, Dojo, Mootools, among others)	Yes (jQuery, Ext JS [https:// en.wikipedia. org/ wiki/Ext_JS], Dojo, YUI, Mootools, among others)
Cloud computing	Yes (Oracle Public Cloud, Oracle Web Logic Server and Google App Engine)	Yes (Amazon EC2, Linode [https:// www.linode. com/], Rackspace [https:// www. rackspace. com/] and Heroku)	Yes (Jelastic [https:// jelastic.com] and Google App Engine)	Yes (Amazon EC2 and Rackspace)	Yes (Cloud Foundry)	Yes (Cloud Foundry, Google App Engine, Amazon EC2 and Heroku)	Yes (dot Cloud, Google App Engine and Amazon EC2)	Yes (Amazon EC2)
Comet support	Yes (ICEfaces [https:// en. wikipedia.	No	No	No	Yes (Comet Actor)	Yes (Atmosphere or CometD [https://	Yes (Orbited)	Yes (Twiggy)

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https:// cakephp.org]	Lift	Grails	Django	Catalyst
	org/wiki/ ICEfaces])					cometd.org/] plugins)		
Custom error messages	Yes (File properties)	Yes (File .yml [https:// en. wikipedia.org/wiki/YAML])	Yes (File properties)	Yes (Model-message)	Yes (Snippet-s.notice, s.error)	Yes (File properties)	Yes (model-validation Error)	Yes (Catalyst::Action:: RenderView:: ErrorHandler)
Customization and extensibility	Yes (PrimeFaces [https:// www. primefaces .org/], RichFaces [https://richfaces.jboss. org/], ICEfaces)	Yes (Plugins e.g. LessCSS [http://lesscss.org/], Authlogic [https://github.com/ binarylogic/ authlogic] among others)	Yes (Plugins, e.g., jsCalendar [https:// gramthanos.github.io/ jsCalendar/], Google Guice [https:// github.com/google/ guice], among others)	Yes (Plugins, e.g., Mandrill, CakeDC [https:// github.com/ CakeDC], among others)	Yes (Modules, e.g., PayPal, Widgets, among others)	Yes (Utility, e.g., iCalendar, Smartionary [https://grails. org/plugin/ smartionary], among others)	Yes (Django-Utilse.g. safestring, translation, among others)	Yes (Plugins-Catalyst::Plugin:: AutoCRUD, among others)
Debugging	Yes (ui:debug)	Yes (debug, to_yamland inspect)	Yes (Struts 2 configuration plugin and debugging interceptor)	Yes (Debug Kit plugin)	Yes (SBT and Maven)	Yes (X-Grails-Resources-Original- Src Header)	Yes (Django Debug Toolbar)	Yes (Komodo)

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https://cakephp.org]	Lift	Grails	Django	Catalyst
Documentation	Yes (Javadoc [https://en.wikipedia.org/wiki/Javadoc])	Yes (Rdoc)	Yes (Javadoc)	Yes (php domain)	Yes (Scaladoc [https://docs. scala-lang.org/style/scaladoc.html])	Yes (grails doc)	Yes (Sphinx)	Yes (Plain Old Documentation, POD)
Forms validation	Yes (JSF standard validators and Bean validation)	Yes (Active Record [https://guides.rubyonrails.org/active_record_basics. html] validations)	Yes (ActionForm)	Yes (Form Helper)	Yes (LiftScreen and Wizard)	Yes (Spring's Validator)	Yes (Django Form)	Yes (HTML::Form-Handler and HTML::Form-Validator)
HTML5 support	Yes (Pass-through attributes)	No	No	No	Yes (HTML5 Properties)	Yes (Modernizr [https://modernizr. com/])	Yes (HTML5 Boilerplate, H5BP)	Yes
Internationalisation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https://cakephp.org]	Lift	Grails	Django	Catalyst
JavaScript framework support	Yes (Dojo, Ext JS, jQuery, among others)	Yes (jQuery, script.aculo.us, Dojo, among others)	Yes (jQuery, Dojo, Ext JS, among others)	Yes (Ext JS, jQuery, Dojo, among others)	Yes (jQuery, script.aculo.us, Ext JS, among others)	Yes (jQuery, Dojo, script.aculo.us, among others)	Yes (jQuery, Ext JS, Dojo, among others)	Yes (jQuery, Dojo, Ext JS, among others)
ORM (object relational mapping)	Yes	Yes (Active Record)	Yes	Yes (Active Record and Data mapper patterns)	Yes (Mapper and Record)	Yes (GORM)	Yes (Django ORM)	Yes (DBIx::Class and Rose::DB::Object)
Parallel rendering	No	No	No	No	Yes	No	No	No
Platform support	Windows (Java Developers Kit(JDK)) - Linux (JDK) - OS X (JDK)	Windows (Rails installer) - Linux (JDK) - OS X (JDK)	Windows (JDK)- Linux (JDK) - OS X (JDK)	Windows (PHP5.2.8 or greater, HTTP Server)-Linux (PHP 5.2.8 or greater, HttpServer) - OS X (PHP 5.2.8 or greater, HttpServer)	Windows (Scala and Java Runtime Environment (JRE)) - Linux (Scala and JRE) - OS X (Scala and JRE)	Windows (JDK and Grails libraries) - Linux (JDK and Grails extensions) - OS X (JDK and Grails libraries)	Windows (Python, Setup tools and PIP) - Linux (Python and PIP) - OS X (Python and PIP)	Windows (perl5.8.6 or higher, Catalyst::Runtime and Catalyst::Devel) - Linux (perl 5.8.6 or higher, Catalyst::Runtime and Catalyst::

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https://cakephp.org]	Lift	Grails	Django	Catalyst
								Devel) - OS X (perl 5.8.6 or higher, Catalyst:: Runtime and Catalyst:: Devel)
REST support	Yes	Yes	Yes	Yes (File config-map Resources)	Yes (Rest Helper)	Yes	Yes (Django REST framework)	Yes (Catalyst:: Controller:: REST)
Scaffolding	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Security	Yes	Yes	Yes	Yes	Yes	Yes (Spring security and Shiro)	Yes	Yes
Site map or automatic menu creation	No	No	No	No	Yes (Site map)	No	Yes (NAV Menu currently being evaluated)	Yes (Menu Grinder)

Best practice	JSF	Ruby on Rails	Struts	CakePHP [https://cakephp.org]	Lift	Grails	Django	Catalyst
Template framework	Yes (Facelets [https://en.wikipedia.org/wiki/Facelets])	Yes	Yes	Yes (Views, elements and layouts)	Yes	Yes (Layout and Site Mesh)	Yes (Django template language)	Yes (Template Toolkit,HTML::Mason,PHP and any extant Perl template engine)
Testing	Yes (JSFUnit [https://jsfunit.jboss.org/])	Yes (Rspec [https://rspec.info/])	Yes (Struts Test Case)	Yes (PHPUnit, Fixtures and Mocking)	Yes (ScalaTest, JUnit, Mocking and Selenium)	Yes (GMock and Easy Mock)	Yes (Unit test, doctest, nose)	Yes (Supports Perl testing standards)
Use actors	No	No	No	No	Yes (Lift Actor)	Yes (GPars)	No	No
Use Lazy loading as part core framework	No	No	No	No	Yes	No	No	No
Use pattern matching	No	Yes	No	No	Yes	No	Yes	Yes
Use Wiring	No	No	No	No	Yes	No	No	No

3.3. MICS web-application architecture

During the web application development process, defining the correct system architecture presents a significant challenge, particularly as the chosen architecture will influence the system maintainability or scalability (Madeyski & Sochmialek, 2005). As part of the definition of the right architecture for the development of the MICS platform the following list of principles (Meier et al., 2008) aims to ensure best practices, promote usability and extendibility, and minimise cost and maintenance requirements. The list will be considered when getting started in the design of the operational prototype and the final system:

- separation of concerns;
- single responsibility principle;
- principle of least knowledge;
- *don't repeat yourself* (DRY);
- avoid doing a big design upfront;
- prefer composition over inheritance.

The development of the MICS platform will aim to implement the MVC framework architecture (Figure 2. Model - View – Controller architecture”), thus reducing the complexity of the web application by separating the logic of the platform into three layers, allowing parallel and autonomous development, potentially using different programming languages. Furthermore, MVC is arguably the most endorsed design pattern, widely becoming the standard in modern software development (Chao et al., 2013) and its implementation in the development of web applications has been well documented by other studies (Majeed & Rauf, 2018; Sarker & Apu, 2014; Pop & Altar, 2014; Leff & Rayfield, 2001), as well as the web development community (Angular[2], Django[3], Laravel[4] and Symfony[5]).

The main value of using MVC architecture in the development of the MICS platform is based on the separation of the model and presentation, as well as, the separation of the view and controller (Madeyski & Sochmialek, 2005). In addition, other advantages (Selfa et al., 2006) with respect to other architectures are:

- less coupling;
- higher cohesion;
- more design clarity;
- facilitated maintenance;
- bigger scalability.

[2]*https://angular.io/*
[3]*https://www.djangoproject.com/*
[4]*https://laravel.com/*
[5]*https://symfony.com/*

3.4. Authentication and authorization system

Users of the MICS platform will be encouraged to use social login through the implementation of **OpenID Connect**[6] as means to access the platform functionalities. This represents one of the most important *single sign-on* (SSO) protocols widely implement for delegate authentication, and it is currently used by Amazon, Google, LinkedIn, and Microsoft amongst others (Mainka et al., 2017). Figure 3. Sign in page for the MICS platform" shows an example of a draft sign in page for the MICS platform.

The implementation of social login within the MICS platform is based on the principle of separation of concerns (see section 3.3. "MICS web-application architecture"). By using OpenID Connect, the MICS platform is aiming to decentralise the user-identity authentication, in addition to comply with article five of the guide to the *general data protection regulation* (GDPR)[7]; consequently, the development will benefit by:

- not having to store user identity information in the application database;
- user's identity being completely separate from the application;
- avoiding developing an authentication and authorization system (development simplicity).

What will be stored about the user is the following information, which will be used to identify returning users within the MICS platform:

SocialMediaID_SocialMedia

For example, **email** or **phone** are the IDs used by Facebook, therefore an instance of stored information for a user logging in via Facebook would be:

aloneinthemyst@yahoo.com_facebook

Security analysis and guidelines with regards to the implementation of OpenID Connect are well documented by different studies (Fett et al., 2017; Li & Mitchell, 2016; Muhammad & Tripathi, 2012). Therefore, and in order to avoid some of these security threats, the development of the MICS platform will be implementing Auth0[8] as authentication and authorization service using the Open Source Program licence, under which the project must comply with the following conditions as stated by the licence:

- the entire codebase must be open source and publicly available on GitHub or similar code hosting services;
- the project cannot charge money for the open source project or any of its derivatives;
- the project must add an Auth0 badge to its website (any badge of the ones shown at [http://auth0.github.io/auth0-oss-badges/]; the code to include can be seen at [https://github.com/auth0/auth0-oss-badges/blob/gh-pages/index.html#L8-L9]);

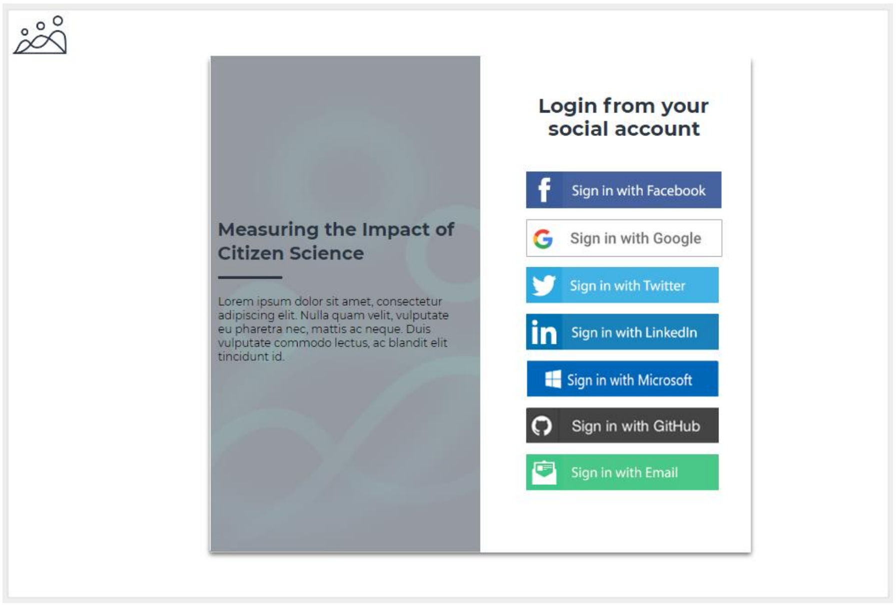
[6]*https://openid.net/connect/*

[7]*The personal data you collect must be limited to what is necessary for processing and must be kept only as long as needed. Appropriate security must be ensured during data processing, including protection against unauthorized or unlawful processing and against accidental loss, destruction, or damage. [https://gdpr.algolia.com/gdpr-article-5].*

[8]*https://auth0.com*

- the *open source software* (OSS) project cannot be deployed to production by another company who will use it to generate revenue.

Figure 3. Sign in page for the MICS platform



Measuring the Impact of Citizen Science

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id.

Login from your social account

3.5. Monitoring and ensuring system performance

Capturing and analysing data aimed to understand the MICS platform behaviour is critical to proactively deal with stability, performance, and anomalies.

Monitoring of the MICS platform will be done by implementing a web *application performance monitoring* (APM) tool; this will provide the data needed to quickly discover, isolate and solve any issues affecting the application's performance, as well as request and response information, and database connection information.

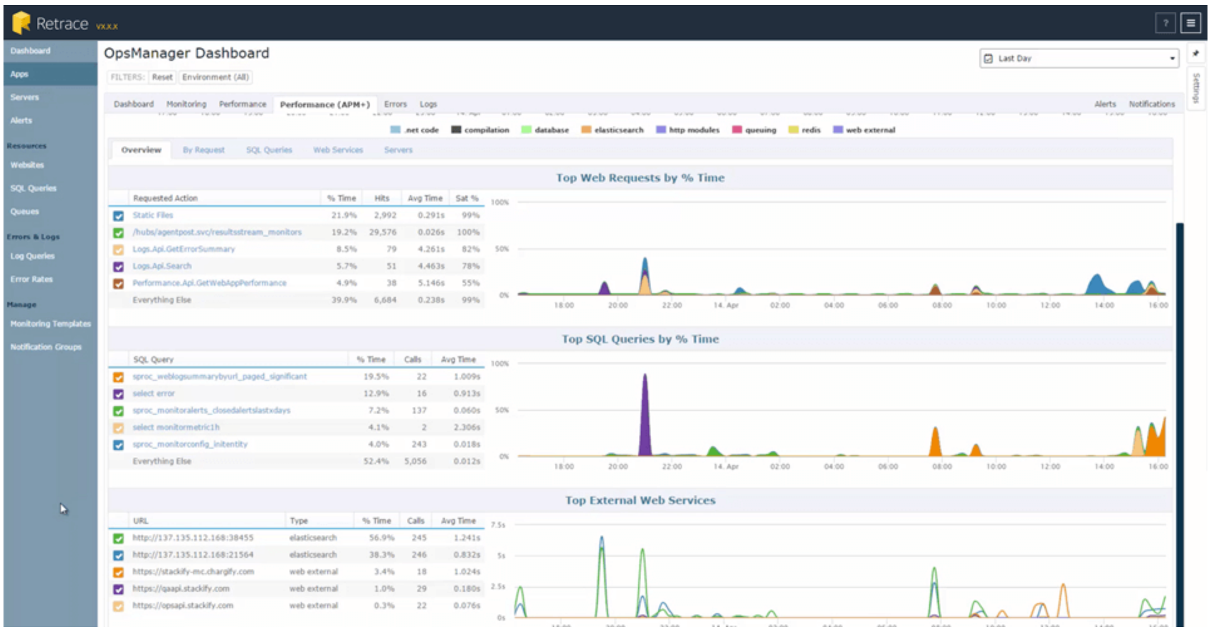
The MICS platform will implement an API-based approach. This provides a programming interface giving the project a level of freedom on how to utilize the APM. Other advantages of this approach include, enabling monitoring components, tracing transactions, and performing error analysis (Rabl et al., 2012).

The development team of the MICS project will select a solution that incorporates as many as possible of the following tools and characteristics:

- application performance monitoring;
- transaction tracing;
- metrics;
- logs;
- errors;
- alerts;
- open source solution.

Currently the MICS project has been offered a 25% discount in the licence for the implementation of a *software as a service* (SaaS) solution named Retrace[9]. Figure 4. Retrace APM dashboard" shows an example of the Retrace dashboard depicting the performance and monitoring graphs for the above-mentioned characteristics.

Figure 4. Retrace APM dashboard



Retrace vx.x.x
Dashboard
Apps
Servers
Alerts
Resources
Websites
SQL Queries
Queues
Errors & Logs
Log Queries
Error Rates
Manage
Monitoring Templates
Notification Groups
OpsManager Dashboard
FILTERS: Reset Environment (All)
Last Day
Dashboard Monitoring Performance Performance (APM+) Errors Logs
Alerts Notifications
.net code compilation database elasticsearch http modules queuing redis web external
Overview By Request SQL Queries Web Services Servers
Top Web Requests by % Time
Top SQL Queries by % Time
Top External Web Services

	Requested Action	% Time	Hits	Avg Time	Sat %
✓	Static Files	21.9%	2,992	0.291s	99%
✓	/hubs/agentpost.svc/resultsstream_monitors	19.2%	29,576	0.026s	100%
✓	Logs.Api.GetErrorSummary	8.5%	79	4.261s	82%
✓	Logs.Api.Search	5.7%	51	4.463s	78%
✓	Performance.Api.GetWebAppPerformance	4.9%	38	5.146s	55%
	Everything Else	39.9%	6,684	0.238s	99%

	SQL Query	% Time	Calls	Avg Time
✓	sproc_weblogsummarybyurl_paged_significant	19.5%	22	1.009s
✓	select error	12.9%	16	0.913s
✓	sproc_monitoralerts_closedalertslastxdays	7.2%	137	0.060s
✓	select monitormetric1h	4.1%	2	2.306s
✓	sproc_monitorconfig_initentity	4.0%	243	0.018s
	Everything Else	52.4%	5,056	0.012s

	URL	Type	% Time	Calls	Avg Time
✓	http://137.135.112.168:38455	elasticsearch	56.9%	245	1.241s
✓	http://137.135.112.168:21564	elasticsearch	38.3%	246	0.832s
✓	https://stackify-mc.chargify.com	web external	3.4%	18	1.024s
✓	https://qaapi.stackify.com	web external	1.0%	29	0.180s
✓	https://opsapi.stackify.com	web external	0.3%	22	0.076s

3.6. MICS assessment tool

The front-end[10] development of the MICS platform will aim to use and adapt, where possible, characteristics and functionalities similar to those of the HEAT and PIA software assessment tools (see Table 1. Assessment tools description") particularly in the area of impact visualisation.

As part of the exploration process for the development of the graphical interface and visualisation as part of the MICS platform, characteristics such as learnability and efficiency were considered whilst assessing the most feasible and suitable group of features (e.g., user interface, data visualisation) and functionalities (e.g., question types, data updates) for the future development of the operational prototype and the final solution of the MICS platform.

Furthermore, **the main principle** taken into consideration whilst prioritising groups of **characteristics and functionalities**, independently of each assessment tool reviewed, was **usability**. This principle, has received great attention, and it is viewed as one significant factor in web application' quality; in addition, this is also considered as a central property when measuring the success of web application development (Abrahão et. al, 2008; Mvungi & Tossy, 2015). The main group of characteristics and functionalities considered were:

[9]*https://stackify.com/retrace/*

[10]*Front-end, also commonly known as the visible part of the website, is everything involved in the graphical interfaces for visualisation and interaction by users using languages such as HTML, CSS and JavaScript.*

- data input requirements;
- clarity of questions and hints;
- design and flow of the different interfaces;
- result presentation – including analytical representations;
- interactivity;
- complexity of the design.

Based on this principle and the characteristics and functionalities mentioned above, the development team of the MICS platform has designed a group of wireframes as first stage in the iterative process for the design of the front-end element of the web application. Further changes and enhancements are expected as part of the development process as result of end users' and consortium's feedback.

3.6.1. Project-data input

Data input for the MICS platform by the user will aim to collect project information needed to measure the impact of the project, and to guarantee interoperability of the platform with existent and future citizen-science initiatives (e.g., SciStarter[11], Zoouniverse[12], CitSci.org[13] and the organisations supporting the Geneva Declaration on Citizen Science Data and Metadata Standards [www.cs-eu.net/news/workshop-report-wg-5-geneva-declaration-citizen-science-data-and-metadata-standards]).

The *European Citizen Science Association* (ECSA) and the COST Action on citizen science[14] have contributed to develop a data model (or ontology)[15] for representing citizen-science projects; thus the MICS platform will be implementing this model as means of data standardisation.

In the design of the platform interface to collect project data, results of existing projects, such as the European project WeObserve [https://www.weobserve.eu/], of which IHE Delft is a partner, will be considered. For example, WeObserve is already using a questionnaire to collect data such as "Geographical Scale", "Stakeholders" or "Sponsor". These concepts will be standardised using the citizen-science ontology, adapted to MICS, integrated into the interface and expanded to cover the scope of the MICS project, which is different from the one of WeObserve.

[11] *https://scistarter.org/*
[12] *https://www.zooniverse.org/*
[13] *https://citsci.org/*
[14] *https://www.cs-eu.net/*
[15] *https://github.com/CitSciAssoc/DMWG-PPSR-Core*

For example, and more specifically, the concept “Geographical Scale” will be formalised using the “GeographicLocation” concept of the ontology (see Figure 5); the concept “Stakeholders” will be formalised using the “Party” and “Participant” concepts of the ontology (see Figure 6 and Figure 7); and the concept “Sponsor” will be formalised using the “FundingProgram” concept of the ontology (see

Figure 8).

Figure 5. “Project.Geography” module of the citizen-science ontology

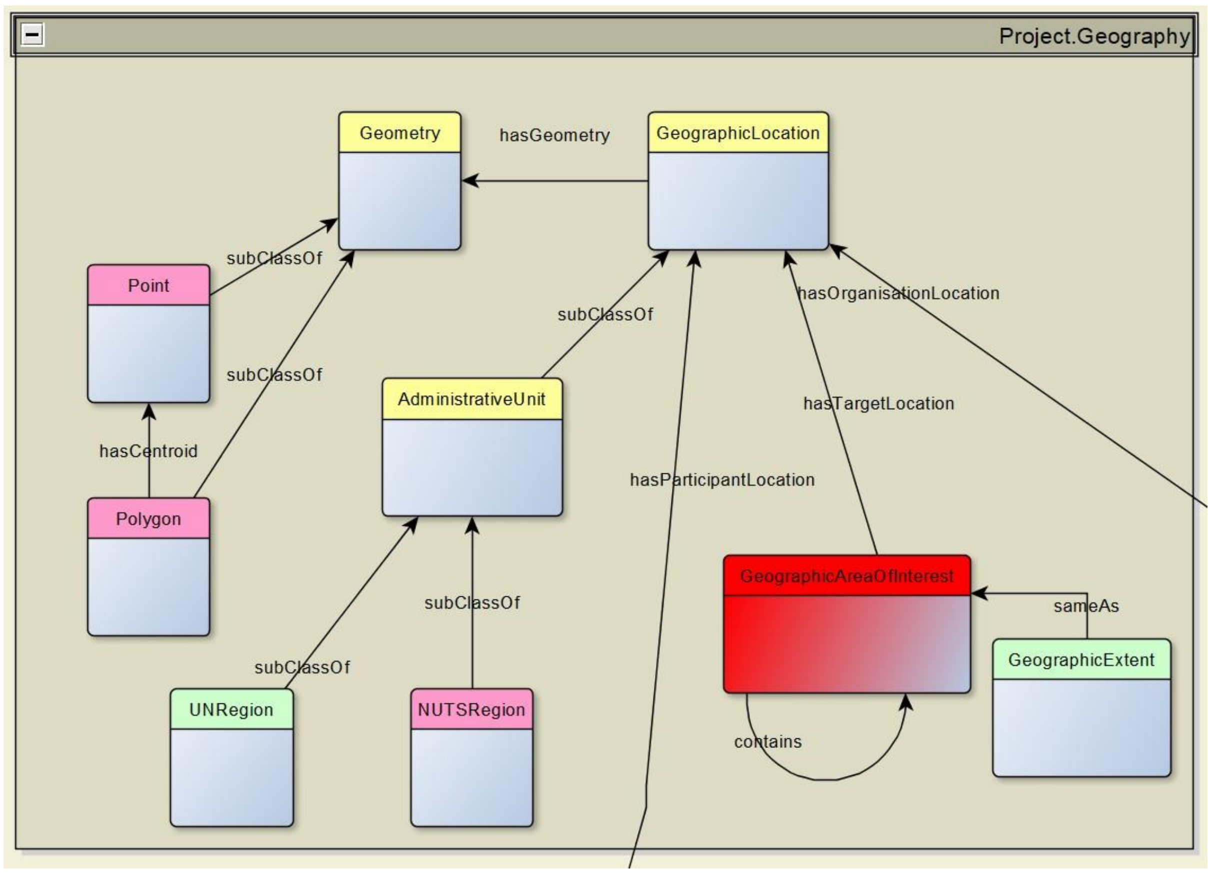


Figure 6. "Project.Affiliates" module of the citizen-science ontology

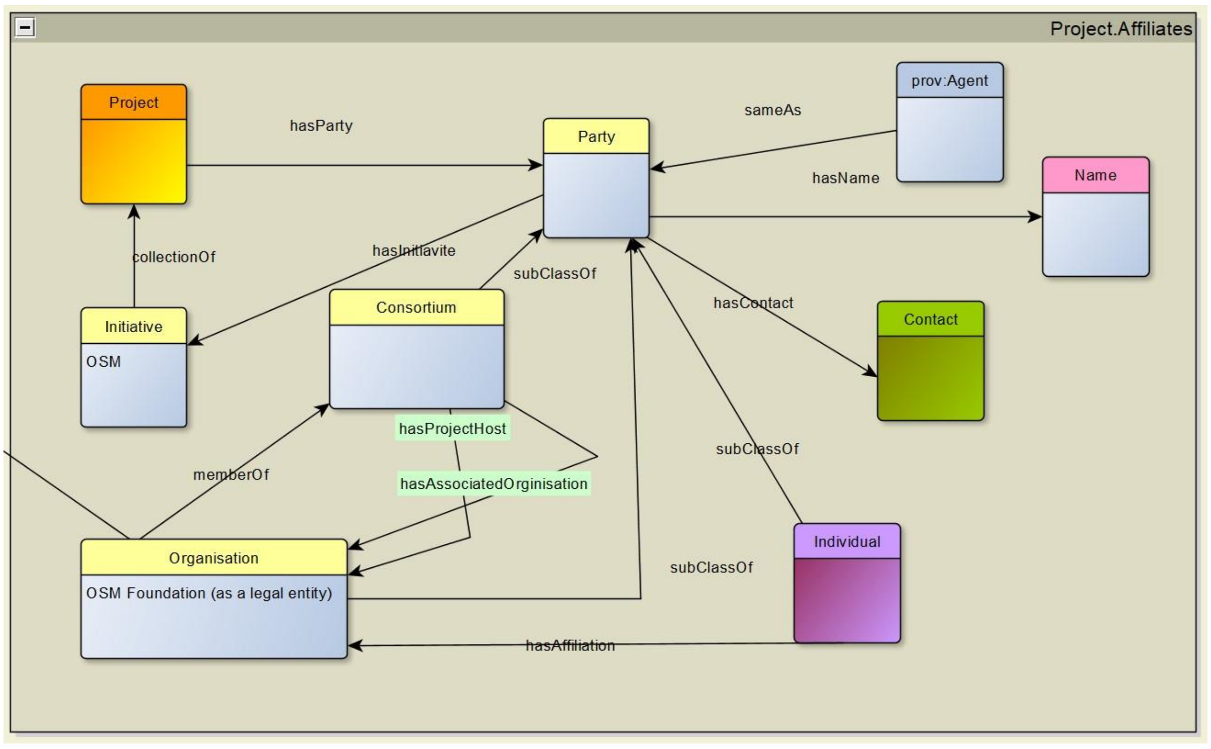


Figure 7. “Project.Participants” module of the citizen-science ontology

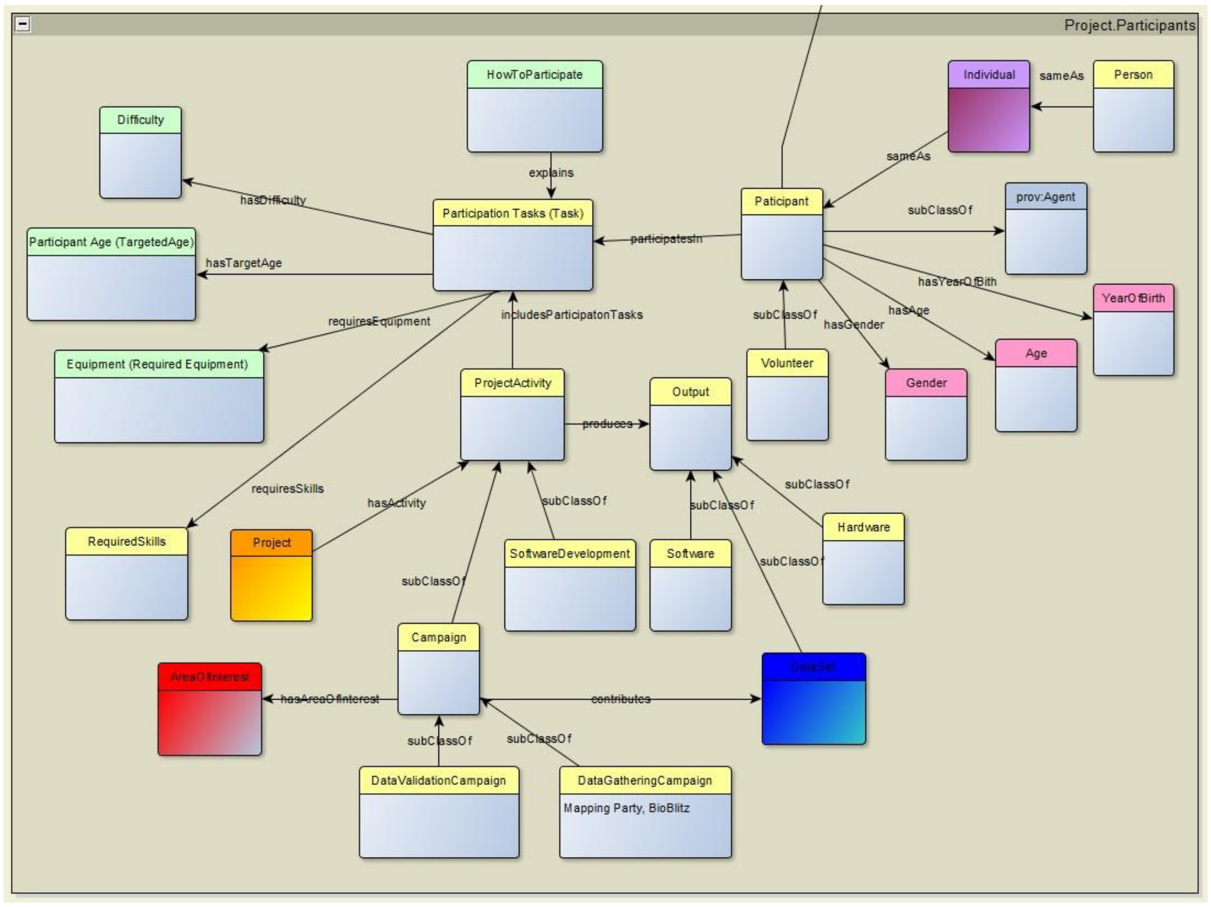


Figure 8. “Project.Funding” module of the citizen-science ontology

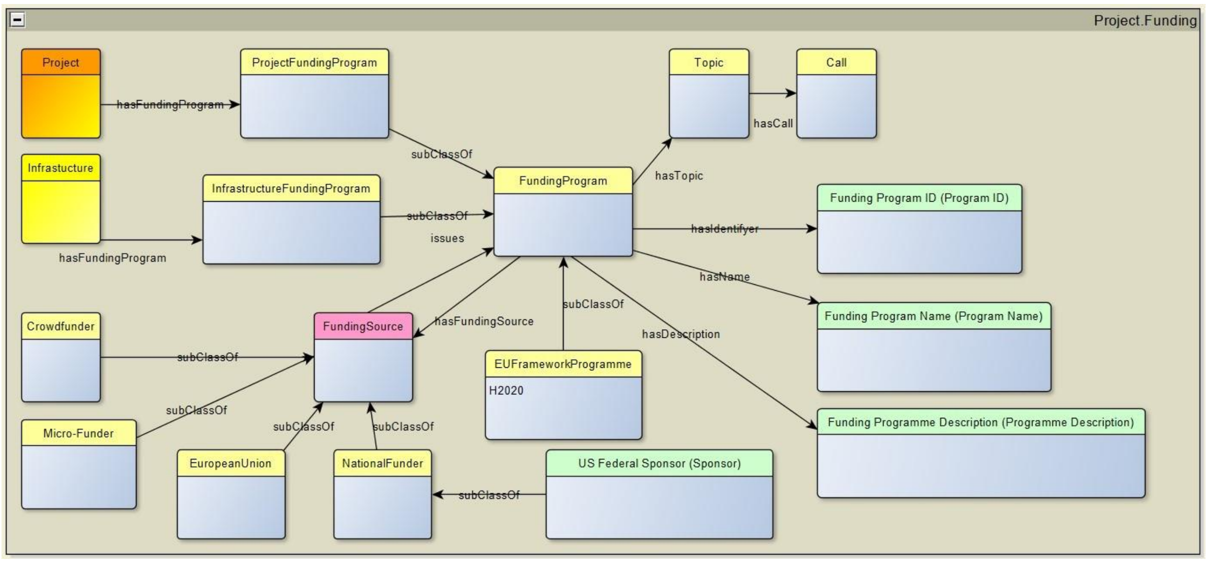
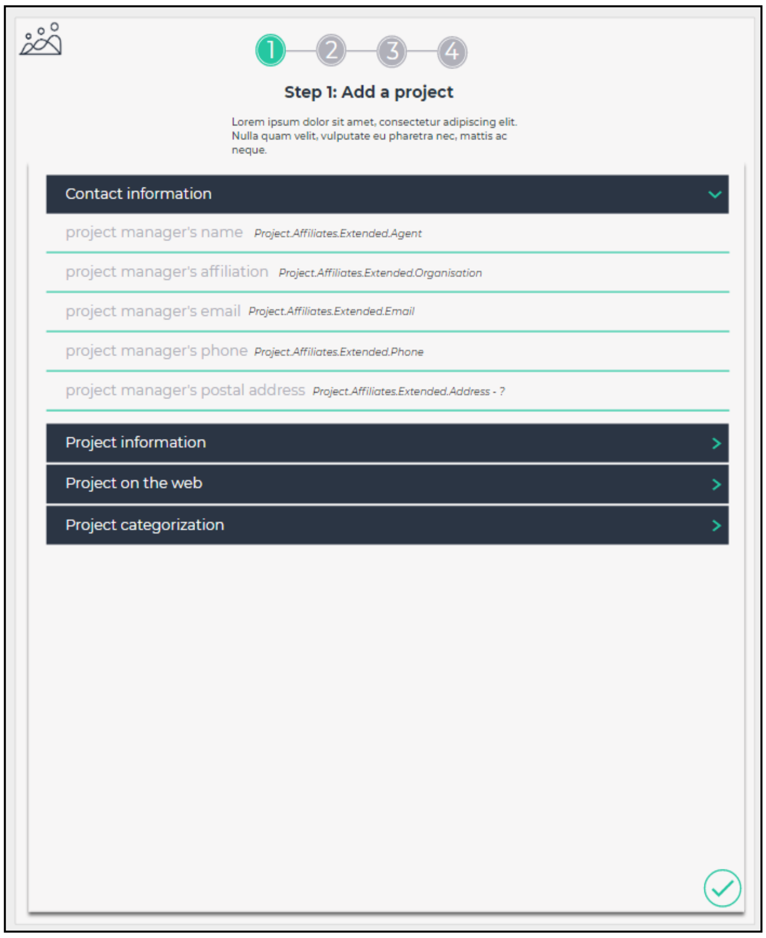


Figure 9. MICS tools - project-data input shows an example of the possible front-end page displaying a predefined set of questions linked to the citizen-science metadata ontology for citizen science projects.

Figure 9. MICS tools - project-data input


1 2 3 4

Step 1: Add a project

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque.

Contact information

project manager's name *Project.Affiliates.Extended.Agent*

project manager's affiliation *Project.Affiliates.Extended.Organisation*

project manager's email *Project.Affiliates.Extended.Email*

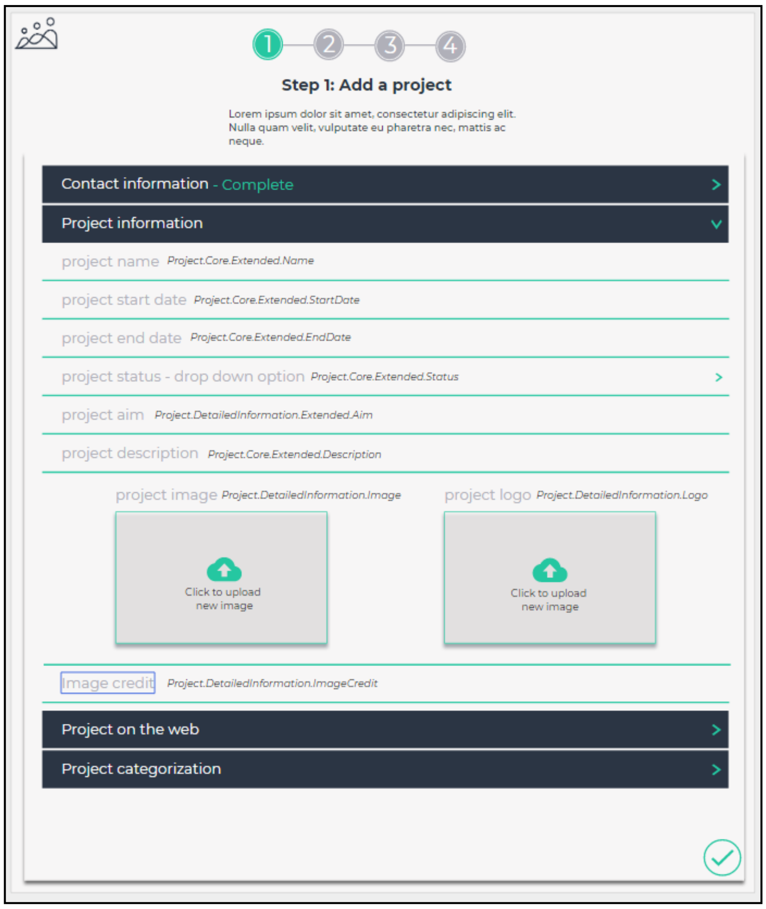
project manager's phone *Project.Affiliates.Extended.Phone*

project manager's postal address *Project.Affiliates.Extended.Address - ?*

Project information

Project on the web

Project categorization



1 2 3 4

Step 1: Add a project

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque.

Contact information - Complete

Project information

project name *Project.Core.Extended.Name*

project start date *Project.Core.Extended.StartDate*

project end date *Project.Core.Extended.EndDate*

project status - drop down option *Project.Core.Extended.Status*

project aim *Project.DetailedInformation.Extended.Aim*

project description *Project.Core.Extended.Description*

project image *Project.DetailedInformation.Image*

Click to upload new image

project logo *Project.DetailedInformation.Logo*

Click to upload new image

Image credit *Project.DetailedInformation.ImageCredit*

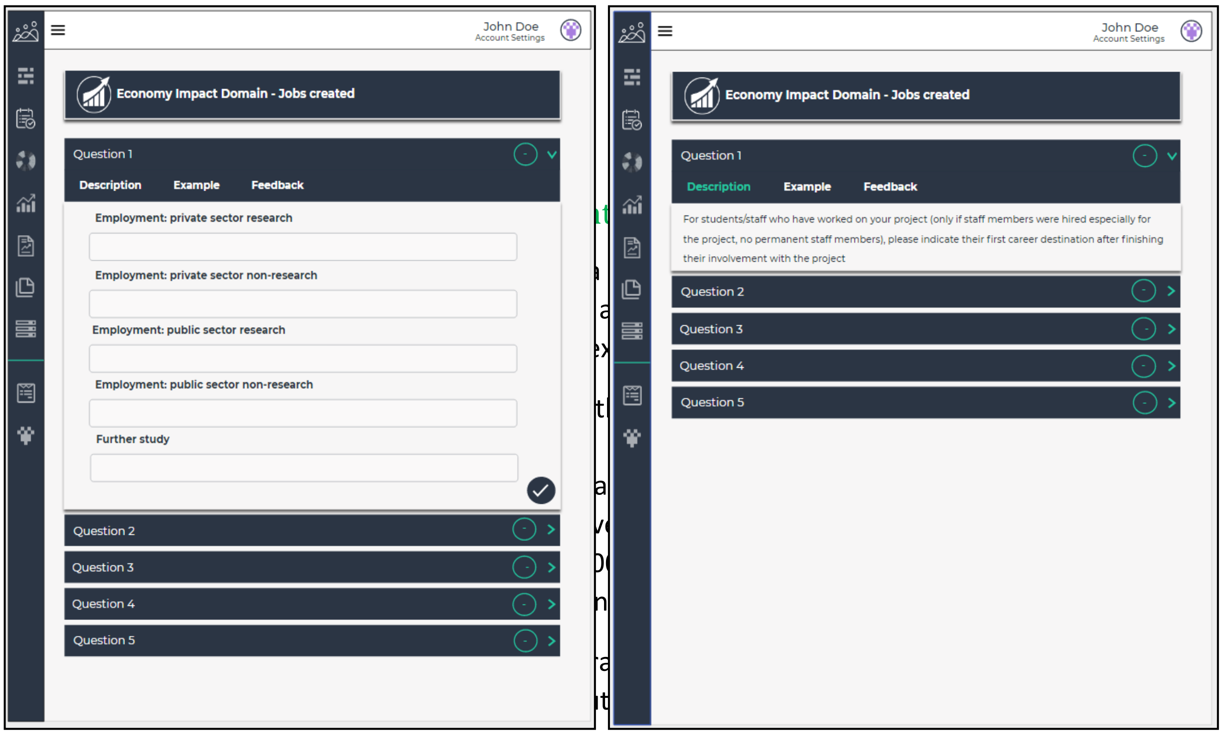
Project on the web

Project categorization

Data input for impact assessment are related to the five domains[16] that form part of the MICS project, as well as, to the set of indicators and metrics defined for each one of these as part of Work Package 2.

Figure 10. MICS tools - project-data input about MICS domains depicts a potential representation of the data input for impact assessment.

Figure 10. MICS tools - project-data input about MICS domains


John Doe
Account Settings

Economy Impact Domain - Jobs created

Question 1

Description Example Feedback

Employment: private sector research

Employment: private sector non-research

Employment: public sector research

Employment: public sector non-research

Further study

Question 2

Question 3

Question 4

Question 5

John Doe
Account Settings

Economy Impact Domain - Jobs created

Question 1

Description Example Feedback

For students/staff who have worked on your project (only if staff members were hired especially for the project, no permanent staff members), please indicate their first career destination after finishing their involvement with the project

Question 2

Question 3

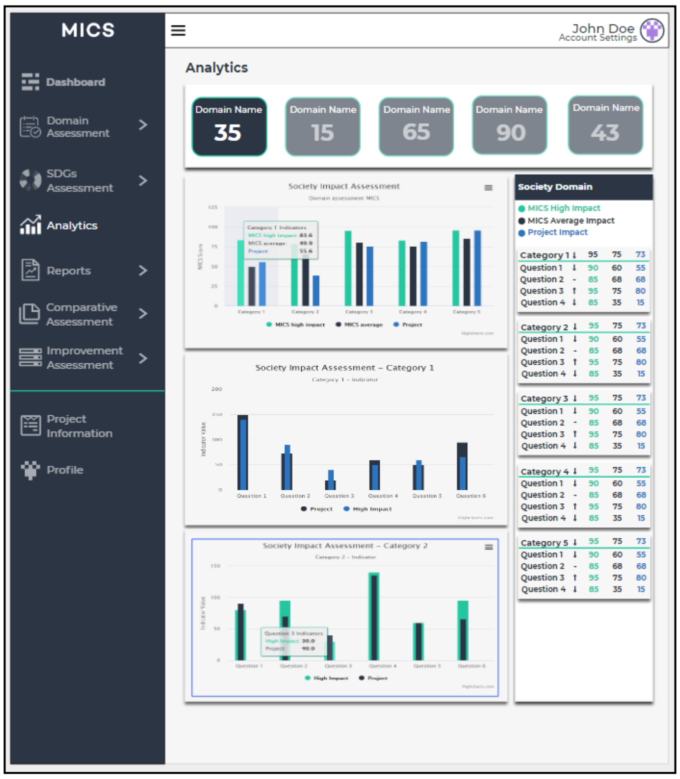
Question 4

Question 5

[16] *Society, governance, economy, environment, and science*

User input will be analysed through an assessment questionnaire against the indicators defined for each domain of the MICS platform. The aim of the assessment analysis is to provide the end user with an interactive and impactful visualisation of the impact assessment of citizen-science projects (see Figure 11. MICS tools - analytics).

Figure 11. MICS tools - analytics



Society Domain

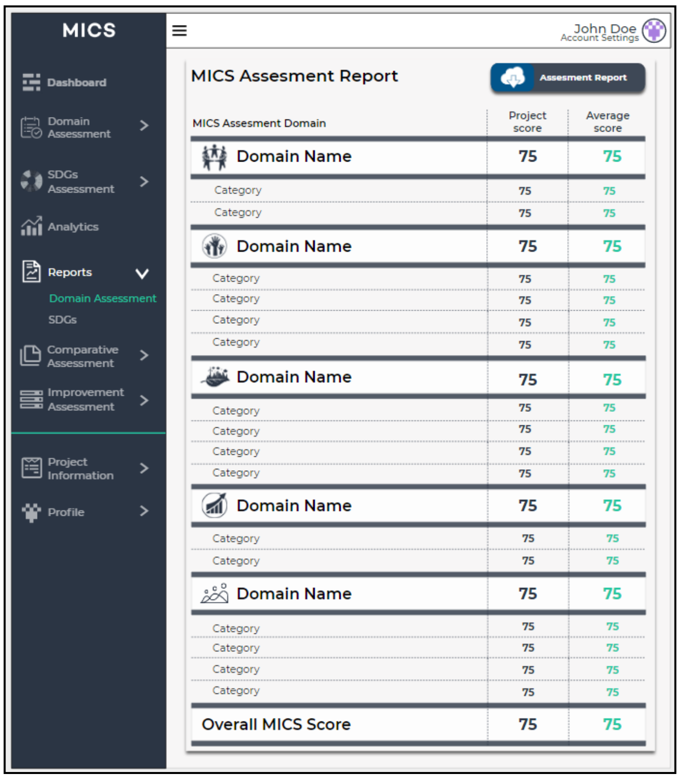
● MICS High Impact
● MICS Average Impact
● Project Impact

Category 1	↓	95	75	73
Question 1	↓	90	60	55
Question 2	-	85	68	68
Question 3	↑	95	75	80
Question 4	↓	85	35	15
Category 2	↓	95	75	73
Question 1	↓	90	60	55
Question 2	-	85	68	68
Question 3	↑	95	75	80
Question 4	↓	85	35	15
Category 3	↓	95	75	73
Question 1	↓	90	60	55
Question 2	-	85	68	68
Question 3	↑	95	75	80
Question 4	↓	85	35	15
Category 4	↓	95	75	73
Question 1	↓	90	60	55
Question 2	-	85	68	68
Question 3	↑	95	75	80
Question 4	↓	85	35	15
Category 5	↓	95	75	73
Question 1	↓	90	60	55
Question 2	-	85	68	68
Question 3	↑	95	75	80
Question 4	↓	85	35	15

A report will represent a condensed version of the score obtained as result of analysing the user's project data input against the MICS's indicators. This type of report (

Figure 12. MICS tools - report) aims to foster the relationship between project managers and citizen scientist by sharing the outcomes of the impact assessment, as well as providing transparency in the application outcomes.

Figure 12. MICS tools - report



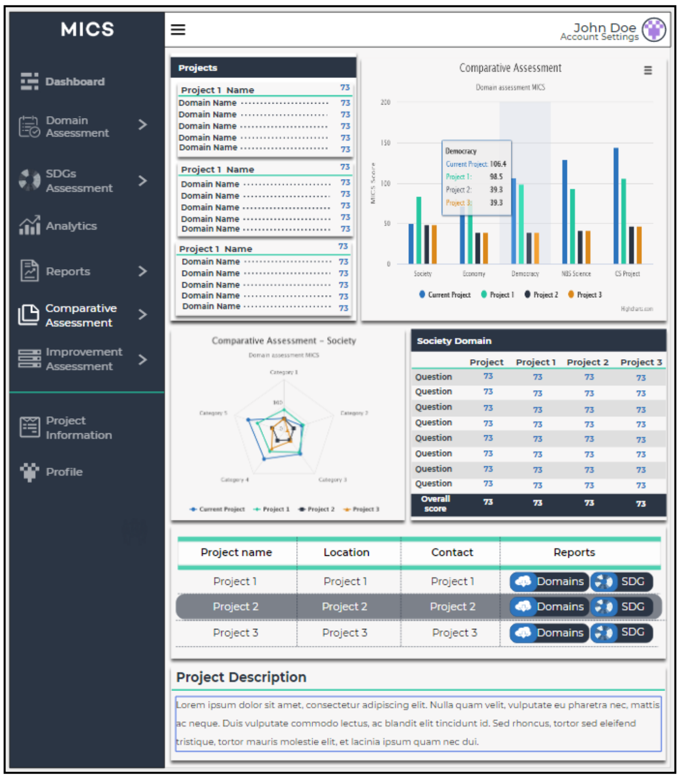
MICS Assesment Report

MICS Assesment Domain	Project score	Average score
Domain Name	75	75
Category	75	75
Category	75	75
Domain Name	75	75
Category	75	75
Category	75	75
Category	75	75
Category	75	75
Domain Name	75	75
Category	75	75
Category	75	75
Category	75	75
Category	75	75
Domain Name	75	75
Category	75	75
Category	75	75
Domain Name	75	75
Category	75	75
Category	75	75
Category	75	75
Category	75	75
Overall MICS Score	75	75

The result of the assessment analysis of an individual project will be compared against the data held by the MICS database. The aim of the comparative analysis (

Figure 13. MICS tools - comparative analysis) is to foster community collaboration between projects using the assessment tool.

Figure 13. MICS tools - comparative analysis



Projects

Project 1 Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73

Project 1 Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73

Project 1 Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73
Domain Name	73

Society Domain

	Project	Project 1	Project 2	Project 3
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Question	73	73	73	73
Overall score	73	73	73	73

Project name	Location	Contact	Reports
Project 1	Project 1	Project 1	Domains SDG
Project 2	Project 2	Project 2	Domains SDG
Project 3	Project 3	Project 3	Domains SDG

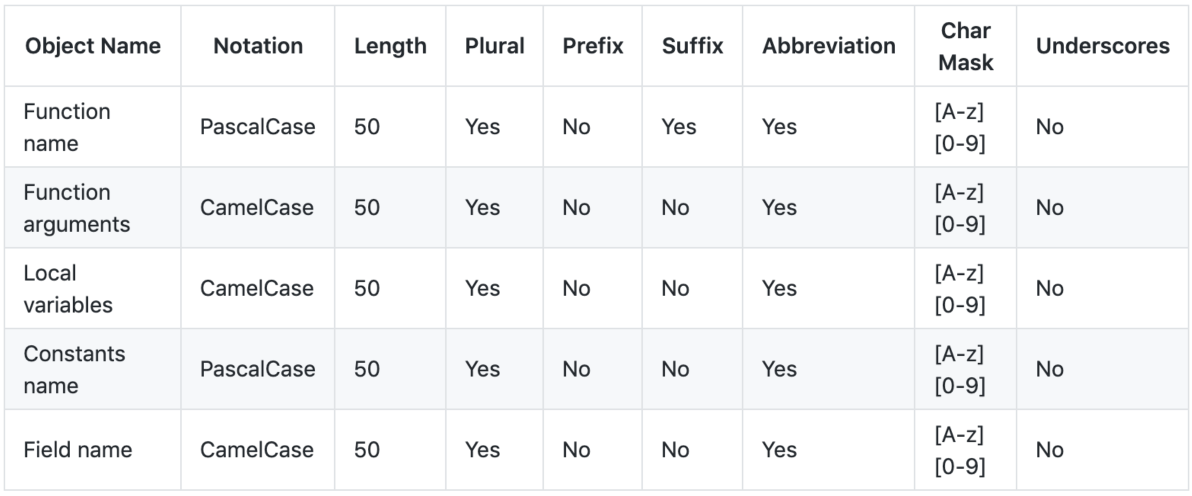
Project Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui.

3.7. Programming standards

Coding standards are a set of industry-recognized best practices that provide a variety of guidelines for developing software code. There is evidence to suggest that compliance to coding standards in software development can enhance team communication, reduce program errors and improve code quality (Li & Prasad, 2005). Coding style guidelines will improve code readability and maintenance, and these cover aspects such as naming and declaration rules for variables and functions, as well as use of white space, indentation, and comments. An example of the type of coding style guidelines to be followed during the development is JavaScript, Figure 14. JavaScript name and coding conventions depicts the JavaScript name and coding conventions that will be implemented during the development of the MICS platform. Further information regarding coding standards for other programming languages as well as relation databases that will be implemented in the development of the MICS platform can be found at [https://github.com/Earthwatch-Institute/naming-convention].

Figure 14. JavaScript name and coding conventions



Object Name	Notation	Length	Plural	Prefix	Suffix	Abbreviation	Char Mask	Underscores
Function name	PascalCase	50	Yes	No	Yes	Yes	[A-z] [0-9]	No
Function arguments	CamelCase	50	Yes	No	No	Yes	[A-z] [0-9]	No
Local variables	CamelCase	50	Yes	No	No	Yes	[A-z] [0-9]	No
Constants name	PascalCase	50	Yes	No	No	Yes	[A-z] [0-9]	No
Field name	CamelCase	50	Yes	No	No	Yes	[A-z] [0-9]	No

Other standards and conventions, which will be considered are:

- Exchange formats: For structure data when possible the preferred format for data exchange and representation in MICS platform will be JSON. Further, XML will be used when specific tool and/or systems will benefit from doing so, i.e. increase in performance, tool optimization.
- Encoding: Documents, tool and subsystem will be constructed to receive and produce UTF-8, hence all textual data in MICS platform is UTF-8.

3.8. Code repository

GitHub is the version control system chosen for the development of the MICS platform. The repository will be created under the Earthwatch institutional account in GitHub and will be made public but only commits from the MICS development team will be accepted into the master branch. The repository URL for the MICS platform will be [https://github.com/Earthwatch-Institute/mics].

3.9. Next steps

The intended users of this document are the software engineers at GeoEcoMar and Earthwatch in charge of developing the MICS platform. Starting from August 2019, and together with the rest of the Consortium, especially WP2 and WP3, they will lead the design of the technological part of the platform and will consequently implement its first prototype, corresponding to deliverable D3.4 "Participatory, adaptive, personalised, information-delivery web platform, period-1 prototype (P1P)" (DEM, PU, M18).

4. *References*

Abrahão, S., Cachero, C., & Matera, M. (2008). Web usability and accessibility. Journal of Web Engineering, 7(4), 257-257.)

Burbeck, S. (1992). Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). Smalltalk-80 v2, 5, 1-11.

Chao, J., Parker, K., & Davey, B. (2013, July). Navigating the framework jungle for teaching web application development. In Proceedings of the Informing Science and Information Technology Education Conference. Informing Science Institute.

del Pilar Salas-Zárate, M., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A., & Cuadrado, J. L. L. (2015). Analyzing best practices on Web development frameworks: The lift approach. Science of Computer Programming, 102, 1-19.

Fett, D., Küsters, R., & Schmitz, G. (2017, August). The web SSO standard openid connect: In-depth formal security analysis and security guidelines. In 2017 IEEE 30th Computer Security Foundations Symposium (CSF) (pp. 189-202). IEEE

Gafni, R., & Nissim, D. (2014). To social login or not login? Exploring factors affecting the decision. Issues in Informing Science and Information Technology, 11, 57-72.

Gordillo, S., Rossi, G., Moreira, A., Araujo, J., Vairetti, C., & Urbieta, M. (2006, October). Modeling and Composing Navigational Concerns in Web Applications. Requirements and Design Issues. In 2006 Fourth Latin American Web Congress (pp. 25-31). IEEE.

Hu, R., Wang, Z., Hu, J., Xu, J., & Xie, J. (2008, October). Agile web development with web framework. In 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing (pp. 1-4). IEEE

Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In Proceedings fifth ieee international enterprise distributed object computing conference (pp. 118-127). IEEE.

Li, W., & Mitchell, C. J. (2016, July). Analysing the Security of Google's implementation of OpenID Connect. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 357-376). Springer, Cham.

Li, X., & Prasad, C. (2005, October). Effectively teaching coding standards in programming. In Proceedings of the 6th conference on Information technology education (pp. 239-244). ACM.

Madeyski, L., & Sochmialek, M. (2005). Architectural design of modern web applications. Foundations of Computing and Decision Sciences, 30(1), 49-60.

Mainka, C., Mladenov, V., Schwenk, J., & Wich, T. (2017, April). SoK: single sign-on security—an evaluation of openID connect. In 2017 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 251-266). IEEE.

Majeed, A., & Rauf I. (2018, September). MVC Architecture: A Detailed Insight to the Modern Web Applications Development. Peer Rev J Sol Photoen Sys .1(1). PRSP.000505. 2018.

Meier, J., Homer, A., Hill, D., Taylor, J., Bansonde, P., Wall, L., Boucher Jr, R. & Bogawat, A. (2008). Web Application Architecture Guide - Application Architecture Pocket Guide Series. [online] Cis.msjc.edu. Available at: http://cis.msjc.edu/CSIS116B/Resources/WebArchitecturePocketGuide.pdf [Accessed 10 May 2019].

Muhammad, A., & Tripathi, N. (2012). Evaluation of OpenID-based double-factor authentication for preventing session hijacking in web applications. Journal of Computers (Finland), 7(11), 2623-2628

Musser, J., & O’reilly, T. (2006). Web 2.0. Principles and Best Practices.[Excerpt]. oO: O'Reilly Media.).

Mvungi, J., & Tossy, T. (2015). Usability evaluation methods and principles for the web. International Journal of Computer Science and Information Security, 13(7), 86.

Naik, N., & Jenkins, P. (2017, May). Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect. In 2017 11th International Conference on Research Challenges in Information Science (RCIS) (pp. 163-174). IEEE.

Plekhanova, J. (2009). Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. Institute for Business and Information Technology.

Pop, D. P., & Altar, A. (2014). Designing an MVC model for rapid web application development. Procedia Engineering, 69, 1172-1179.

Prokofyeva, N., & Boltunova, V. (2017). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. Procedia Computer Science, 104, 51-56.

Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H. A., & Mankovskii, S. (2012). Solving big data challenges for enterprise application performance management. *Proceedings of the VLDB Endowment*, *5*(12), 1724-1735.

Reenskaug, T., & Coplien, J. (2013). More deeply, the framework exists to separate the representation of information from user interaction. The DCI Architecture: A New Vision of Object-Oriented Programming.

Sarker, I. H., & Apu, K. (2014). MVC architecture driven design and implementation of java framework for developing desktop application. International Journal of Hybrid Information Technology, 7(5), 317-322.

Selfa, D. M., Carrillo, M., & Boone, M. D. R. (2006, February). A database and web application based on MVC architecture. In 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06) (pp. 48-48). IEEE.