bEhaVioral Insights anD Effective eNergy policy acTions

Project No. 957117

Project acronym: EVIDENT

Project title:

Behavioural Insights and Effective Energy Policy Actions

Deliverable 6.5

Verification and Validation Report for final version of EVIDENT platform

Programme: H2020-LC-SC3-EE-2020-1

Start date of project: December 01, 2020

Duration: 36 months

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957117

Document Control Page

Deliverable Name	Verification and Validation
Deliverable Number	D6.5
Work Package	WP6 Prototyping and integration
Associated Task	T6.3 System Integration, Verification and Validation
Covered Period	M25 (December 2022) - M39 (February 2024)
Due Date	February 29, 2024
Completion Date	February 27, 2024
Submission Date	February 29, 2024
Deliverable Lead Partner	CERTH
Deliverable Author(s)	Christos Ntoumanopoulos (CERTH), George Sidiras (CERTH), Paris Karypidis (DUTH), Athanasios Tziouvaras (Bi2S), Panagiotis Sarigiannidis (UOWM), Georgios Karagiannidis (UOWM), Stamatia Bibi (UOWM), Georgios Fragulis (UOWM), Anna Triantafyllou (UOWM), Athanasios Liatifis (UOWM), Dimitrios Pliatsios (UOWM), Vasileios Melissianos (PPC), Eleftherios Fountoukidis (SID), Thomai Karamitsou (SID), Anastasios Lytos (SID), Theocharis Saoulidis (SID), Konstantinos Kyranou (SID), Zisis Batzos (SID), Georgios Michoulis (SID)
Version	1

Dissemination Level		
PU	Public	**X**
CO	Confidential to a group specified by the consortium (including the Commission Services)	

Document History

Version	Date	Change History	Author(s)	Organisation
0.1	September 19, 2023	ToC Formulation	George Sidiras	CERTH
0.2	November 12, 2023	Initial version of Section 4	Paris Karypidis, Vasileios Melissianos, Athanasios Tziouvaras	DUTH, PPC, Bi2S
0.3	December 27, 2023	Initial version of Section 5	Paris Karypidis, Christos Ntoumanopoulos	DUTH, CERTH
0.4	December 27, 2023	Final version of Sections 4 and 5	Paris Karypidis	DUTH
0.5	January 25, 2024	Initial version of Section 2	Christos Ntoumanopoulos, Panagiotis Sarigiannidis, Georgios Karagiannidis, Stamatia Bibi, Georgios	CERTH, UOWM

			Fragulis, Anna Triantafyllou, Athanasios Liatifis, Dimitrios Pliatsios	
0.6	January 28, 2024	Initial version of Section 3	Christos Ntoumanopoulos	CERTH
0.7	February 5, 2024	Initial version of Section 6	Christos Ntoumanopoulos, Eleftherios Fountoukidis, Thomai Karamitsou, Anastasios Lytos, Theocharis Saoulidis, Konstantinos Kyranou, Zisis Batzos, Georgios Michoulis	CERTH, SID
0.8	February 10, 2024	Finalize all sections of the deliverable, including Section 1 and Section 6	Christos Ntoumanopoulos	CERTH
1.0	February 27, 2024	Final Version of the Deliverable after Review	Christos Ntoumanopoulos	CERTH

Internal Review History

Name	Institution	Date
Konstantinos Kyranou	SID	25 February 2024
Ioannis Neokosmidis	Bi2S	25 February 2024

Quality Manager Revision

Name	Institution	Date
Dimosthenis Ioannidis	CERTH	27 February 2024

Table of Contents

List of Figures

List of Tables

List of Code Blocks

List of Acronyms

Acronym	Explanation
AAA	Arrange-Act-Assert
API	Application Programming Interface
CD	Continues Development
CI	Continues Integration
DX.Y	Deliverable X.Y
MVT	Model-View Template
PoC	Proof of Concept
QUIS	Questionnaire for User Interface Satisfaction
SD	Standard Deviation
TDD	Test-Driven Development
TX.Y	Task X.Y
UAT	User Acceptance Testing
URS	User Requirements Specification
UxEM	Experience Evaluation Metrics
VM	Virtual Machine

Executive Summary

In the proof of concept (PoC) for the EVIDENT platform, the pilot design plays a crucial role in validating the platform's capabilities and assessing its effectiveness within real-world scenarios. The design phase ensures that the project meets the specific requirements of the uses of the platform and that the integration needs with external systems are covered as well. The PoC is a crucial step for validating the platform's feasibility and integration capabilities before proceeding to full-scale implementation.

Within this work, the consortium leverages methodologies such as agile development and DevOps and uses them together to enhance the success of PoC, by enabling fast integration and validation of concepts; while also fostering a collaborative environment for continuous learning and improvement. Validation is conducted though comprehensive testing encompassing functional performance and security aspects. Thus, the EVIDENT platform can be properly evaluated to ensure reliability, functionality, and security. Such testing efforts contribute to building user confidence by identifying any issues and ensuring overall quality.

1. Introduction

Agile methodologies such as Scrum or Kanban enable teams to break down the PoC into smaller and manageable iterations allowing for fast feedback and learning. The same principle works with DevOps practices such as continuous integration and delivery. These practices enable the fast validation of concepts and faster interaction between developers. Likewise, agile and DevOps promote collaboration and communication among team members, stakeholders, and users. This collaborative environment encourages knowledge sharing, aligns expectations and ensures that everyone is involved through the PoC.

Verification is the process of evaluating the software to ensure that it meets the specified requirements, designs and standards. In case of a new software version, verification involves checking whether the new version aligns with the planned features, enhancements, or bug fixes. It may include unit testing to verify the implementation is aligned with coding standards and functions. Validation is the process under which the software determines whether it satisfies the users' needs and expectations. It involves user acceptance tests (UAT) and functional testing to validate that the software behaves as expected and meets the users' requirements. The integration process refers to combining different software modules or systems into a unified and functioning application. The integration ensures the new version can communicate well with the rest of the system, and all components are implemented without compatibility issues. Furthermore, the verification, validation, and integration may overlap during the software development process. This often happens in order to make sure that new software versions are technically correct (verification) and meet user expectations (validation) while also being seamlessly integrated with the rest of the system.

1.1 Purpose of the Deliverable

This document elaborates on the complex validation and verification procedures applied in the EVIDENT platform. It includes both the theoretical foundations and the way they were implemented. It provides a solid conceptual foundation that supports the platform's operational structure by going into the core ideas of validation and verification. Moreover, it explains the distinctive validation and verification procedures the EVIDENT platform uses, bridging theory and practice with ease. These include a wide range of procedures, such as unit testing, continuous integration processes, and the validation of external platform components, such as the EVIDENT serious game. In addition, it emphasises the importance of user acceptability, engagement, and feedback by providing thorough plans that are appropriate for both technical and non-technical users. Finally, it provides researchers with simple, understandable instructions for accessing the EVIDENT website. This includes an in-depth explanation of the platform's architecture, exact directions for doing research, and comprehensive advice on integrating the serious gaming component. In the end, it aims to promote user-friendliness, trustworthiness, and transparency within the EVIDENT platform's validation and verification architecture.

1.2 Relation with other Deliverables and Tasks

Deliverable 6.5 is the output of Task 6.3 "System Integration, Verification and Validation". Task 6.3 receives D6.1 "Architecture, design and integration documentation", D6.2 "Crowdsourcing Tools of EVIDENT platform", D6.3 "Gamification Tools of EVIDENT platform" and D6.4 "Datahub Services of EVIDENT platform". Those Deliverables are related to Task 6.1 "System Architecture and Design

Specifications" and Task 6.2 "System Development". Moreover, these tasks include the prototyping of the EVIDENT platform based on inputs from WP1-WP4 and early findings from WP5.

1.3 Structure of the Document

This deliverable is organised as follows:

- Section 2 – Verification Process: Presents the fundamental concepts of validation and verification, their function, and how those terms are applied to the platform's validation and verification process in theory.
- Section 3 – Validation and Verification Testing of Proof of Concept: Describes the EVIDENT platform's verification and validation processes through the use of unit tests, continuous integration, and the validation of an external platform component, such as the EVIDENT's serious game.
- Section 4 – User acceptance and verification: Presents the user's acceptance of the platform and their opinion, with the analysis of questionnaires for technical and non-technical users.
- Section 5 – EVIDENT platform training materials: Provides an overview of the EVIDENT platform's structure and instructions on how users may access it, do research, and, if needed, include a serious game in their study.
- Section 6 – Performance and Scalability Assessment: Demonstrates EVIDENT platform's continuous integration and deployment process. It also presents the methods employed by the consortium to publicise the platform and attract new users.
- Section 7 – Conclusion: This section summarises the deliverable.

2. Verification Process

2.1 Requirements Verification

Platform verification, which aims to ensure the accuracy, dependability, and security of digital platforms, is a crucial part of software development and system engineering. In the following subsections, the consortium goes into the foundational facets of the platform's verification, including its goals, approaches, difficulties, and best practices.

2.1.1 The role of Verification

Verification is the methodical process of determining if a digital platform complies with predetermined requirements and quality standards. It sets itself apart from validation, which is concerned with determining if the platform satisfies user needs.

Verification is crucial for a number of reasons, including:

- **Building Trust:** By confirming that a platform works as planned and is free of errors, verification helps users develop trust.
- **Compliance:** To fulfil compliance duties, it is necessary to conduct verification in many sectors due to regulatory regulations and standards.
- **Security:** In order to find and fix security flaws and protect sensitive data and systems, verification is crucial.

2.1.2 Key Aspects of Platform Verification

Platform verification encompasses several critical aspects:

- **Functional Verification:** This process ensures that the platform's features and functions work as intended.
- **Performance Verification:** Evaluate the platform's performance under various loads to make sure it can scale and that its resources are being used effectively.
- **Security Verification:** Focuses on locating and fixing security flaws, defending against online attacks, and preventing unwanted access.
- **Compliance Verification:** Assures that the platform complies with rules and guidelines relevant to the industry, particularly in fields like healthcare, banking, and aerospace.

2.1.3 Verification Methodologies

Verification encompasses both manual and automated techniques:

- **Manual Verification:** Uses human expertise to examine code, run tests, and verify compliance. Code reviews, walkthroughs, and manual testing are a few examples.
- **Automated Verification:** Modern platform verification relies heavily on automation. Effective verification procedures depend on continuous integration (CI), continuous deployment (CD), and automated testing pipelines. Automation facilitates quick feedback loops and early issue discovery in addition to speeding up verification.

Moreover, verification is given mathematical rigour by formal procedures like model checking and theorem proving. These methods entail building formal models of a platform and analysing it

mathematically to check its qualities. For critical systems where accuracy is crucial, formal verification is very beneficial.

2.1.4 Challenges in Platform Verification

Platform verification presents several challenges:

- **Complexity:** Complete verification is a challenging undertaking because of the complexity of today's digital systems, which include various sophisticated parts and interconnections.
- **Scalability:** Ensuring that verification grows and adapts to the platform's size and complexity is an ongoing task.
- **Security Landscapes:** Verification must keep up with emerging vulnerabilities and attack vectors as cyber threats develop.

2.1.5 Best Practices in Platform Verification

Effective platform verification requires adherence to best practices, including:

- **Early and Continuous Verification:** Early project verification and ongoing verification throughout the development lifecycle.
- **Test-Driven Development (TDD):** Writing tests before developing code allows test-driven development (TDD) to direct development and guarantee testability.
- **Security by Design:** Security by Design refers to the process of designing the platform with security in mind, as opposed to considering security as an afterthought.
- **Automation:** To increase productivity and lower human error, use automation for testing, deployment, and compliance checks.

2.2 Requirements Validation

A platform's validation is a crucial step in ensuring its efficacy and integrity. It entails a thorough assessment covering a range of factors, such as functionality, performance, security, usability, and compliance. An overview of the platform’s validation essential steps is provided in this section.

2.2.1 Objectives for Validation

The platform validation procedure's primary objectives are to:

- **Confirm Compliance**: Ensure that the platform conforms with established legal duties, industry-specific standards, and regulatory restrictions. For the platform to comply with legal obligations and preserve its operational integrity, it must follow regulatory frameworks and standards unique to the sector that adheres to. Validation's primary goal is compliance since it protects both the platform and its users.
- **Verify Functionality**: Verify that the platform's features and capabilities comply with the stated requirements and pre-set criteria. Validation's functional component makes sure the platform serves its intended purpose. To ensure appropriate operation, it covers a range of testing depths, from unit testing to system-wide evaluations.
- **Assess Security**: By finding weaknesses and potential threats using testing approaches like penetration testing and vulnerability scanning, the consortium evaluates the platform's security

measures. Security is a top priority in a time when cyber threats are always changing. The platform's security measures are extensively evaluated during the validation phase to guard it against any weaknesses.

- **Enhance Usability**: Make sure the platform meets user expectations for interface and usability by providing an intuitive and friendly experience. Usability testing is concerned with the user experience and works to make the platform simple to use. This factor is essential for customer satisfaction and wider uptake.

2.2.2 Validation Plan and methodology

2.2.2.1 Validation Plan

The validation plan acts as the overall procedure's theoretical road map. It provides an organised approach to validation efforts by defining the scope, methodology, resource allocation, and timeline of the whole process.

A thorough document that presents a schedule for the full validation process is also known as the validation plan. It describes the parameters, goals, and particular tasks to be carried out during validation. It also analyses the required resources and specifies the duties and functions of the validation team.

Organizations may make sure that validation efforts are systematic, well-organized, and in line with their goals by defining a clear validation plan. This strategy serves as a foundation for execution, thus making sure that all necessary actions are completed to validate the platform successfully.

2.2.2.2 Methodology

The validation methodology comprises four pillars:

- **Functional Testing:** Unit, integration, and system testing are used to determine whether the platform complies with predetermined requirements. Functional testing is concerned with evaluating the platform's functionality to make sure it complies with the requirements. This entails unit testing of specific parts, integration testing of the interactions between different modules, and system testing of the overall functionality of the system. Confirming that the platform works as intended and that all features are functional is the main objective.
- **Performance Testing:** Examines the responsiveness of the platform under various circumstances, such as load, stress, and scalability testing. To understand how the platform functions in various scenarios, performance testing is crucial. Stress testing pushes the platform to its limits to find potential bottlenecks; load testing evaluates its capability to handle changing levels of user activity; and scalability testing gauges its ability to expand with rising usage. This pillar makes sure that the platform can produce an excellent level of performance in actual use.
- **Security Testing:** Vulnerabilities are addressed by vulnerability scanning and penetration testing. In order to find and reduce potential security hazards, security testing is essential. While penetration testing simulates actual attacks to find vulnerabilities that could be used by criminal actors, vulnerability scanning entails searching the platform for known flaws. This pillar makes sure the platform is safe and capable of safeguarding users and sensitive data.
- **Usability Testing:** Ensures that the interface and user-friendliness meet user expectations. In order to determine how easy it is for consumers to interact with the platform, usability testing concentrates on the user experience. This involves assessing the user-friendliness of the

platform's interface, navigation, and overall design. Ensuring the platform is user-friendly and fits their demands is the main objective.

2.2.2.3 Timeline

Time management in theory is essential. The timetable, which includes deadlines and milestones guarantees a well-planned and speedy validation process.
Planning ahead is essential in validation attempts. Testing, documenting, and reporting are just a few of the validation tasks that are scheduled in the timeline. This task also contains benchmarks that denote significant accomplishments and dates that guarantee the validation process goes along as intended. A clearly established timeline aids in retaining concentration and completing validation goals within allotted time constraints.

2.2.3 Practical Application

2.2.3.1 User Requirements Specification (URS)

The URS document captures both functional and non-functional requirements specified by end users and stakeholders, bridging user expectations and validation processes. Thus, tt acts as a bridge between the validation processes and user expectations. Both functional and non-functional needs as identified by end users and stakeholders are fully included in the URS document. These specifications serve as the foundation for validation testing, which makes sure that the platform complies with user expectations and is used as intended.

2.2.3.2 Functional Testing

Through unit, integration, and system testing, functional testing verifies compliance with requirements. A crucial step in the validation process is functional testing, which is concerned with verifying that the platform's functionality complies with the given requirements. It involves various testing levels:

- **Unit Testing:** In this type of testing, the platform's individual parts or units are examined independently. It aims to confirm that every component performs as intended.
- **Integration Testing:** Integration testing evaluates how various platform modules or components interact with one another. It makes sure that there are no conflicts in how these parts function together.
- **System Testing:** System testing looks at the platform as a whole and assesses its overall performance and functionality. It confirms that every aspect of the system functions as intended and satisfies user needs.

Functional testing is crucial for locating and fixing any functional inconsistencies or problems that might appear while a platform is being used.

2.2.3.3 Performance Testing

Performance testing, which includes load, stress, and scalability testing, evaluates responsiveness under various conditions. Performance testing is a crucial component of validation because it evaluates how the platform operates in various scenarios. It contains:

- **Load Testing:** Load testing measures how well a platform performs when subjected to varying levels of user activity. It ensures that the platform maintains responsiveness and determines how it manages increased user loads.

- **Stress Testing:** Stress testing involves pushing a platform to its breaking point in order to find any potential bottlenecks or weak points. In order to determine the platform's breaking point, the process simulates extremely harsh conditions.
- **Scalability Testing:** Scalability tests gauge a platform's capacity to grow and accommodate more users over time. It guarantees that the platform can expand to meet user demand.

In order to ensure optimal performance, performance testing identifies areas for development and offers insightful information about the platform's capabilities.

2.2.3.4 Security Testing

Through penetration testing and vulnerability scanning, security testing resolves vulnerabilities. In the current digital environment, security is of the utmost importance. A thorough process called security testing seeks to spot and reduce potential security vulnerabilities. It contains:

- **Vulnerability Scanning:** Vulnerability scanning entails looking for known weaknesses and vulnerabilities on the platform. It aids in locating places/modules that might be attacked.
- **Penetration Testing:** Penetration testing mimics actual attacks on the platform to find vulnerabilities that could be used by hostile actors. It strengthens the platform's security precautions.

The platform is put through security testing to make sure it is safe from threats and that sensitive data and user information are protected.

2.2.3.5 Usability Testing

Usability testing guarantees that the interface and user-friendliness are in line with user expectations. User happiness is significantly influenced by usability. To make sure that the platform meets user expectations, usability testing evaluates the user interface and overall user experience. It consists of:

- **Interface Evaluation:** Usability testing assesses the platform's interface for intuitiveness, clarity, and navigability.
- **User-Friendliness Assessment:** This evaluates the platform's overall user-friendliness, including how simple it is for users to carry out tasks and accomplish their objectives.

Usability testing strives to improve the user experience and make the platform more usable and pleasurable for users.

2.2.3.6 Validation Maintenance

The platform's validation status is maintained through continual monitoring and upgrades, and periodic revalidation ensures continuing compliance and optimization.

Validation requires continual commitment rather than a one-time effort. The platform is continuously monitored as part of validation maintenance to make sure it is still secure, compliant, and performing at its best. Regular revalidation checks are carried out to verify continuing compliance and find any fresh problems that can emerge as the platform develops.

3. Validation and Verification Testing of Proof of Concept

In this section, the validation and the verification process used on the Evident Platform are explained. The more important steps for the validation process are a series of unit tests to test all the functionalities of the platform and the validation of the data transfer between the platform and the EVIDENT's Serious Game, which is the only other external component of the EVIDENT project, except the EVIDENT platform itself.

3.1 Unit Tests of the Evident Platform

A crucial step in the software verification process is unit testing, which involves meticulously examining each unit or component of a software program separately. Depending on the design of the software, these units might be functions, classes, modules, or even full services. The importance of unit testing, its benefits, and the guidelines for creating strong unit tests are all covered in this section.

3.1.1 The Role of Unit Testing

In order to guarantee the quality and dependability of software, unit testing is critical. Unit testing is essential for the following reasons:

1. **Early Defect Detection**: One of the primary benefits of unit testing is its ability to detect defects early in the development cycle. Developers can find flaws and fix them by carefully examining isolated pieces of code before they get more complicated and expensive.
2. **Improved Code Quality**: Unit testing encourages developers to write code that is modular, maintainable, and robust. When code is designed with testability in mind, it tends to be of higher quality, reducing the possibility of bugs and making it easier to maintain and enhance over time.
3. **Documentation and Code Understanding**: Individual code modules' anticipated behaviour is documented using unit tests. For both present and future developers working on the project, they offer insights into how a certain piece of code is meant to function.
4. **Regression Testing**: A safety net is provided by unit tests when the codebase is altered. Developers may quickly identify regressions with the use of a complete set of unit tests, guaranteeing that new code changes do not impair previously functioning code.

3.1.2 Principals for Effective Unit Tests

As unit testing is a crucial part of the software development process, it is important that all unit tests are completed and passed before committing code into repositories. Specifically, a single test should be implemented for each major method, function, class, or internal Application Programming Interface (API) of each enabler's component. Test inputs and outputs should be deterministic, have clear and unambiguous error messages, and have a one-to-one relationship with the functionality being tested. Some universal testing principles also apply, such as:

1. **Test One Thing at a Time**: The primary goal of each unit test should be to validate a particular piece of functionality. To retain precision and clarity, avoid mixing numerous features or situations into a single test instance.

2. **Descriptive Test Names**: Select names for your tests that are meaningful and descriptive. An effective test name should make clear what is being tested and the desired result, making it simpler to comprehend and manage.
3. **The Arrange-Act-Assert (AAA) Pattern**: Utilize the AAA pattern to structure your unit tests:
    3.1. **Arrange**: Create the test environment's objects, initialize its variables, and set up any necessary pre-requisites.
    3.2. **Act**: Put into practice the particular feature or technique being examined.
    3.3. **Assert**: Confirm that the action's outcome corresponds to what was anticipated.
4. **Maintainability**: Treat unit tests as though they were production code. Keep tests clean, legible, and manageable by updating them frequently to reflect changes in the source.
5. **Independency**: The test should be designed to run independently of other tests in a clean environment and before the main code is invoked. Moreover, tests should be able to run in random order.
6. **Automation**: The best strategy is to have the tests automated, without that meaning that manual tests cannot exist.
7. **Failure**: Well-designed tests should be able to fail, meaning that by changing the input, the test may not always pass.

3.1.3 Unit Tests and Continuous Integration

Unit tests are a crucial factor in ensuring the Continuous Integration of the project. Running unit tests after every update to the platform is a crucial step in maintaining the software's quality and dependability. Developers may quickly find any bugs or regressions that may appear with the new version by automating this procedure and including it in the platform's deployment workflow. This proactive approach to testing not only increases end-users' trust in the stability of the platform but also makes it possible to find and fix bugs quickly, providing a smoother and more reliable experience overall.

3.1.4 EVIDENT platform unit tests

In the EVIDNET platform, there is a total of 306 tests linked with validation of the platform’s requirements. Those requirements are enlisted in Table 1, which is an update for the same table in D6.2, and used as a guidebook for the development of all applications of the EVIDENT platform.

Table 1. Functional and non-functional requirements the unit tests are based on.

Requirement ID	Requirement title	Coverage
Func_01	User Authentication	Implemented 100%. Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions [5].
Func_02	User Authorisation	Implemented 100%. Django authentication system covers the needs of authorisation determining what an authenticated user is allowed to do [5].

Func_03	Organisation Validation	Implemented 100%. Every request for an organisation account is automatically sent to the platform's administrators.
Func_04	Organisation grant access	Implemented 100%. When the request is reviewed by the platform's administrators an email is sent confirming the rights of the user as an organisation.
Func_05	User login	Implemented 100%. The users can login after the platform's authentication/authorisation.
Func_06	User roles	Implemented 100%. The EVIDENT platform provides different roles for the users (super-admin, organiser, participant).
Func_07	User logs out	Implemented 100%. The EVIDENT platform offers the log-out functionality to its logged-in users independently from their role.
Func_08	Complete profile information	Implemented 100%. The users can provide their profile information on their registration.
Func_09	Update profile information	Implemented 100%. The users can provide their profile information on demand.
Func_10	Update user demographics	Implemented 100%. The users can update additional information regarding her/his demographic details.
Func_11	Delete account	Implemented 100%. The user can delete her/his account
Func_12	Forgot password	Implemented 100%. The platform provides the option "forgot password" to the users.
Func_13	Update password	Implemented 100%. The users can update their passwords.
Func_14	Export user profile data	Implemented 100%. The user can export her/his activity from the platform.
Func_15	Create survey	Implemented 100%. The users registered as an organisation can create surveys.
Func_16	Update survey	Implemented 100%. The user/organisation who already created a survey can update it on request.
Func_17	Delete survey	Implemented 100%. The user/organisation can delete the survey he/she created.

Func_18	Overview of existing serious games	Implemented 100%. The user/organisation can overview an existing serious game.
Func_19	Upload a new serious game	Implemented 100%. The user/organisation can upload a serious game.
Func_20	Update a serious game	Implemented 100%. The user/organisation can upload an updated version of a serious game.
Func_21	Overview of existing lab experiments	Implemented 100%. The user/organisation can overview an existing lab experiment.
Func_22	Create a lab experiment	Implemented 100%. The user/organisation can create a new lab experiment.
Func_23	Setup lab experiment wizard	Implemented 100%. The lab experiment wizard shows up when the user creates a new lab experiment.
Func_24	Edit an existing lab experiment	Implemented 100%. The user/organisation can edit existing lab experiments.
Func_25	Delete an existing lab experiment	Implemented 100%. The user/organisation can delete existing lab experiments.
Func_26	Serious game manager	Implemented 100%. The user/organisation can manage Serious Games.
Func_27	Configure a serious game	Implemented 100%. The user/organisation can create a configuration file.
Func_28	Attach a serious game to lab experiment	Implemented 100%. The user/organisation can attach a serious game to each lab experiment.
Func_29	Participate in a lab experiment	Implemented 100%. The user/participant can participate in published lab experiments.
Func_30	Inspect participation in a lab experiment	Implemented 100%. The user/participant can review her/his answers on a lab experiment previously participated.
Func_31	Export answers from a lab experiment	Implemented 100%. The user/organisation that has set up and published a lab experiment, has the option to export the answers to the lab experiment.
Func_32	Share a lab experiment	Implemented 100%. The user/participant that has completed her/his participation in a lab experiment has the option to share it with her friends via social media.

Func_33	User scoreboard	After finalizing the game, the player is prompted with a percentage that indicates their score. The requirement has 100% implemented.
Non_Func_01	Password encryption	All the user's personal data is securely encrypted in accordance to GDPR.
Non_Func_02	Data anonymisation	All the user's personal data is anonymized in to GDPR.
Non_Func_03	Availability	The platform is hosted on a virtual machine (VM) at UOWM premises, and takes all the possible measurements and safeguards for its constant operation.
Non_Func_04	Scalability	The platform is designed in a way that new functions can be added easily by using Django model-view-template (MVT) framework.
Non_Func_05	Reliability	The platform is hosted on a VM at UOWM, and takes all the possible measurements and safeguards for its reliable operation.
Non_Func_06	Recoverability	Due to the GitHub usage, there is always the ability to recover older versions of the platform.
Non_Func_07	Maintainability	With the verification and validation processes described in this deliverable the maintainability of the platform is achieved.
Non_Func_08	Serviceability	The platform is designed in a way to improve the user's experience.

Appendix 1: EVIDENT Platform unit tests log is a preview of the output of all the unit tests run on the platform.

3.1.5 Serious game integration

The only external component that is not a part of the EVIDENT platform is the Serious Game. The integration between the Serious game and the EVIDENT platform is explained in more detail in Section 5 of this deliverable, which can be used as a guide for creating and uploading a game on the platform. More specifically, in subsections 5.3.2.1 and 5.3.2.3 the integration between the platform and the game is described, where the platform sends data to the game and the game sends data to the platform. In this procedure, it was crucial to create an automated series of actions to check that the game is receiving the right information. The serious game is configured based on a configuration file, which was explained in deliverable D2.3 - "Serious game implementation design". This configuration file is depicted in Code block 1.

Configuration file of Serious Game

```
{
        "section1": {
                "age": 30,
                "role": ["Home owner", "Tenant", "Landlord"],
                "income": 10000,
                "gender": "female",
                "familyStatus": "TwoOrMoreChildrens"
        },
        "section2": {
                " frequentBreakdown": true,
                " frequentBreakdownValue": "Normally",
                "dishwasherBreakdown": true,
                "dishwasherValue": "Normally",
                "fridgeBreakdown": true,
                "fridgeValue": "Normally",
                "washingMachineBreakdown": true,
                "washingMachineValue": "Normally",
                "electricStoveBreakdown": true,
                "electricStoveValue": "Normally",
        },
        "language": "en",
        "device": "desktop",
        "debug": "true",
}
```

Code block 1: Configuration file of Serious Game

In this file, there is a debug parameter. If the debug function is true, the game instantly returns the same file to the platform after reading it. With this function, the platform checks if the returned values are the same and the integration process is implemented. Moreover, the game returns some standard values for the JSON file with the user's answers, to check if these answers are in the correct form.

4. User acceptance and verification

As a final verification step for the verification of the EVIDENT platform, we designed and implemented user acceptance testing (UAT). The EVIDENT platform was built to assist the data collection process for the EVIDENT consortium while providing an advanced, free-of-charge solution for the design and implementation of online lab experiments. Thus, user interface and user experience are crucial for the platform's adoption from 3rd party researchers and research institutions. Through the UAT, we ensure that the platform meets all the requirements and that the EVIDENT consortium delivers a high-quality product to the research community. UAT is achieved through two distinct questionnaires:

- The user experience evaluation metrics (UxEM) questionnaire is scenario-based testing where the users should implement the corresponding scenarios and provide feedback on whether they were able to successfully implement them. For the EVIDENT platform, a UxEM questionnaire consisting of 8 different scenarios was created. The participants are requested to perform simple and more complicated tasks, such as registering to the platform and implementing a simple session. The UxEM questionnaire prepared to evaluate EVIDENT's platform use experience can be found in Appendix 9.2.
- The questionnaire for user interface satisfaction (QUIS) [1] is a well-known tool developed to assess the subjective satisfaction of the users on human-computer interface on the overall system satisfaction and interface factors such as screen factors, terminology and system feedback, learning factors and system capabilities. The QUIS questionnaire used for the evaluation of the user interface satisfaction can be found in Appendix 9.3.

For the creation of both questionnaires, Google Forms were leveraged. Both questionnaires were communicated to several people including consortium members and, at the same time, partners and colleagues from sister projects. The participants were selected based on the fact that the EVIDENT consists of diversified consortium, with partners with different backgrounds and expertise. For example, the three university partners (DUTH, TCD, and UOWM) do not have strong expertise in software development and could provide objective feedback about the use of the EVIDENT platform as simple users. Moreover, project partners such as the TCD and JRC are potential users of the EVIDENT user, since a lot of experts in these institutions already work with data collection tools such as Qualtrics [2], SurveyMonkey [3], GoogleForms [4] and others. In addition, the sister projects form a very useful pool of potential users for the EVIDENT platform. Thus, their subjective opinion would be very useful.

4.1 Results of the UxEM and QUIS

This section presents the results of the UxEM questionnaire. It total, 14 replies were collected from both consortium members and sister project partners. As can be seen in Figure 1 and Figure 2, around 57% of the participants have a background in behavioural sciences and economics, while participants' mean value of computer knowledge (self-disclosed) is 4,42M (0,72 SD). That means that participants come from research areas where lab experiments are the main tools for data collection and could be potential users of the EVIDENT platform.

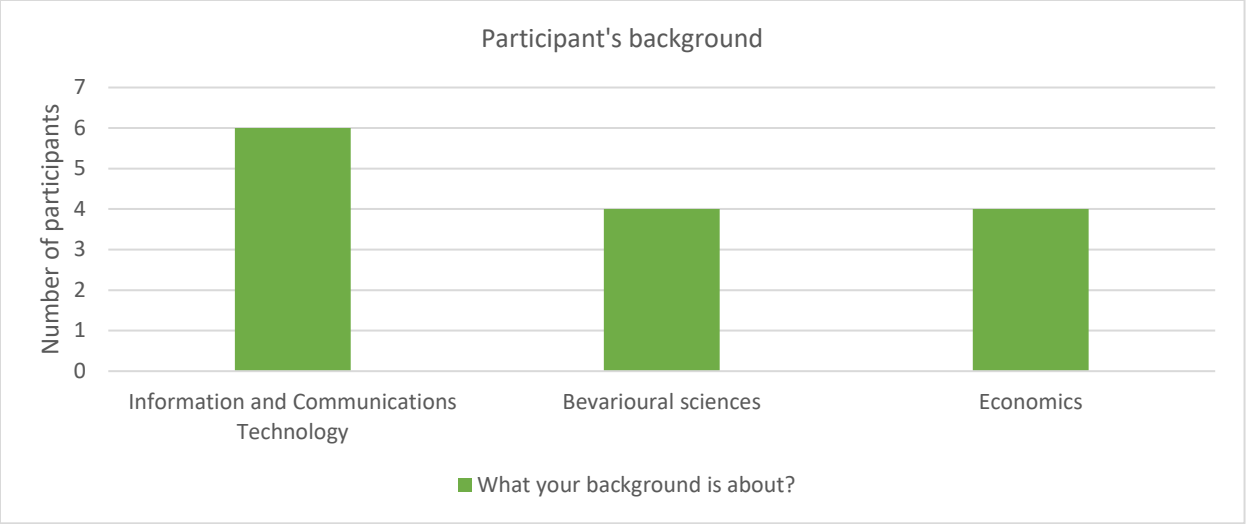


Figure 1: UxEM and QUIS participants' background

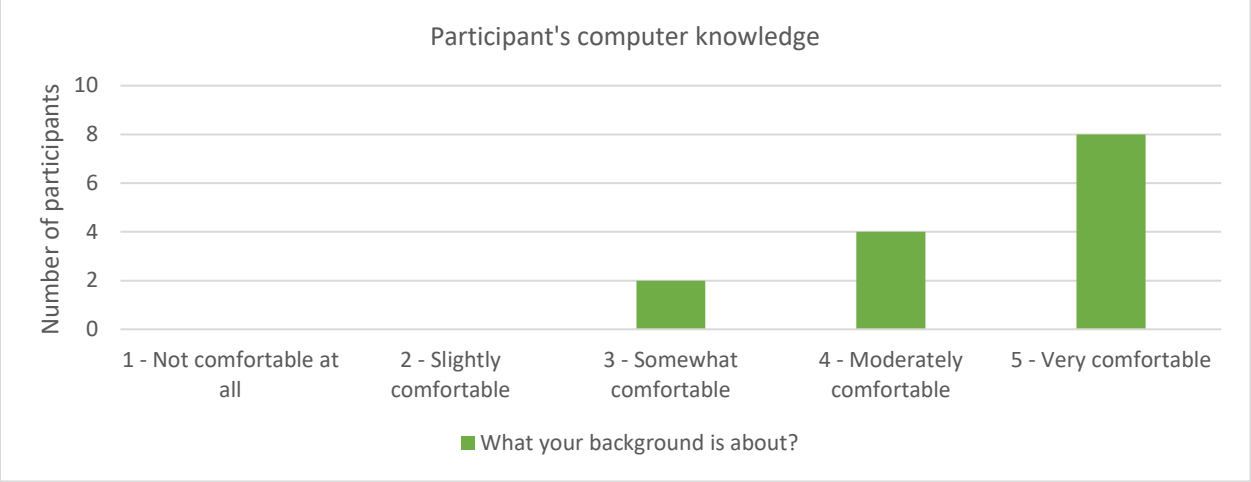


Figure 2: UxEM and QUIS participants' computer knowledge

Regarding the platform's acceptability, we included two additional questions in the QUIS questionnaire asking the participants how likely it would be to a) use the platform for their research and b) propose the platform as a data collection solution to their colleagues.

In both questions, the results are satisfactorily based on the time spent by the participants on the platform. Figure 3 presents the results regarding the potential use of the platform by the participants while Figure 4 illustrates the results when the participants were asked to propose the platform to their colleagues.

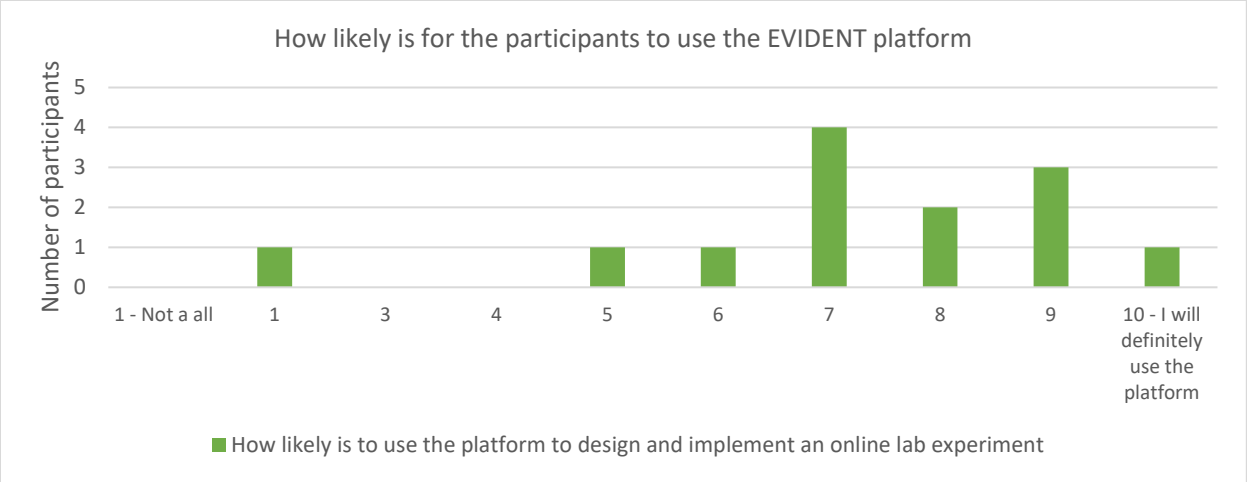


Figure 3: Participants' willingness to use the EVIDENT platform

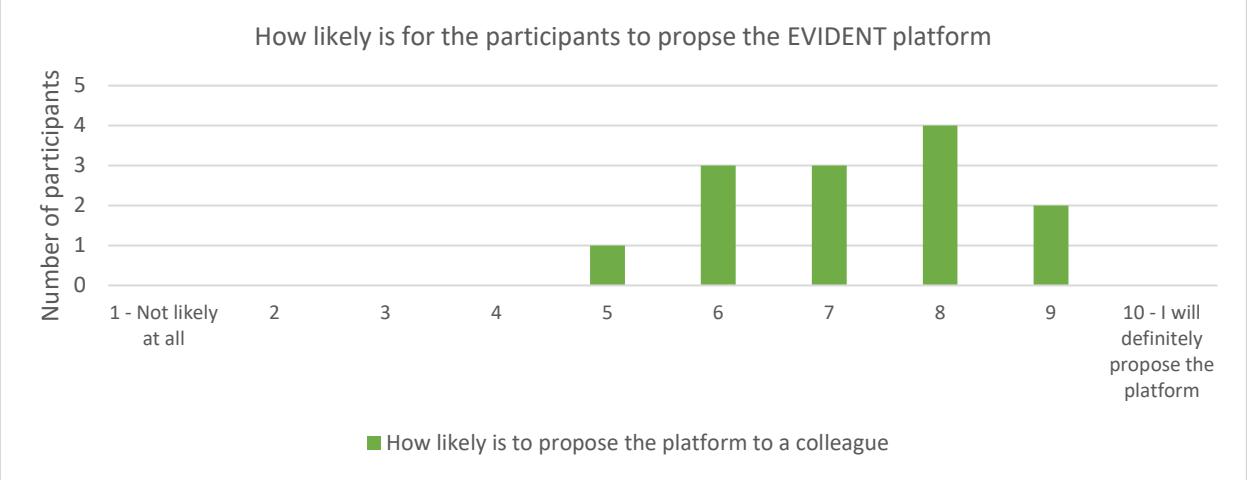


Figure 4: Participants' willingness to propose the EVIDENT platform

4.1.1 User experience evaluation metrics questionnaire results

The user experience evaluation metrics questionnaire is a scenario-based process where the participants are required to fulfil some simple tasks on the EVIDENT platform and at the same time record the time needed and the clicks pressed to complete. Table 2, presents the completion rate of each task along with the average number of time needed and the clicks pressed. Appendix 9.4 presents the corresponding replies collected for each participant.

Table 2. Completion rate, average time and clicks needed for the UxEM.

Scenario	Completion Ratio (%)	Average time in seconds (SD)	Average clicks needed (SD)
Scenario 1: Please, register to the EVIDENT platform as an "organisation".	100	60 (68,89)	5,78 (4,29)
Scenario 2: While logged in, fill in your de-mographics.	100	65,42 (66,42)	21 (8,94)
Scenario 3: While logged in, find a text that describes how your data are used in the EVIDENT platform.	100	39 (40,33)	4,35 (1,54)

Scenario 4: While logged in, find the page that describes how you can use the EVIDENT platform to design and implement a lab experiment.	100	30,14 (26,85)	4,14 (3,29)
Scenario 5: While logged in, create and save a simple questionnaire using the corresponding editor.	78	93,14 (67,87)	31,78 (21,25)
Scenario 6: While logged in, publish your questionnaire by creating a new session.	85	80,07 (54,66)	-
Scenario 7: While logged in, copy your session url and by using an incognito browser fill it in.	92	21,14 (13,51)	-
Scenario 8: While logged in, download the data collected in xlsx format.	92	27,78 (23,88)	4,07 (2,46)

Based on the results in the UxEM, it's clear that most of the participants managed to successfully fulfil the corresponding tasks and use the EVIDENT platform in an intuitive way. Since the EVIDENT platform is built as an ecosystem to provide services to the research community, user experience should be the top priority.

4.1.2 Questionnaire for user interface satisfaction results

This section presents the results of the questionnaire for user interface satisfaction. The EVIDENT platform has achieved a great score, regarding the overall acceptance of the interface satisfaction. That means that the platform was designed to cover a plethora of needs and even assist non-technical users. Regarding the six topics in the QUIS test, the EVIDENT platform reached an average score of 8,18 (1,30 SD). Figure 5 presents the average scores for each QUIS topic while Appendix 9.5 presents the scores for each one of the QUIS answers.

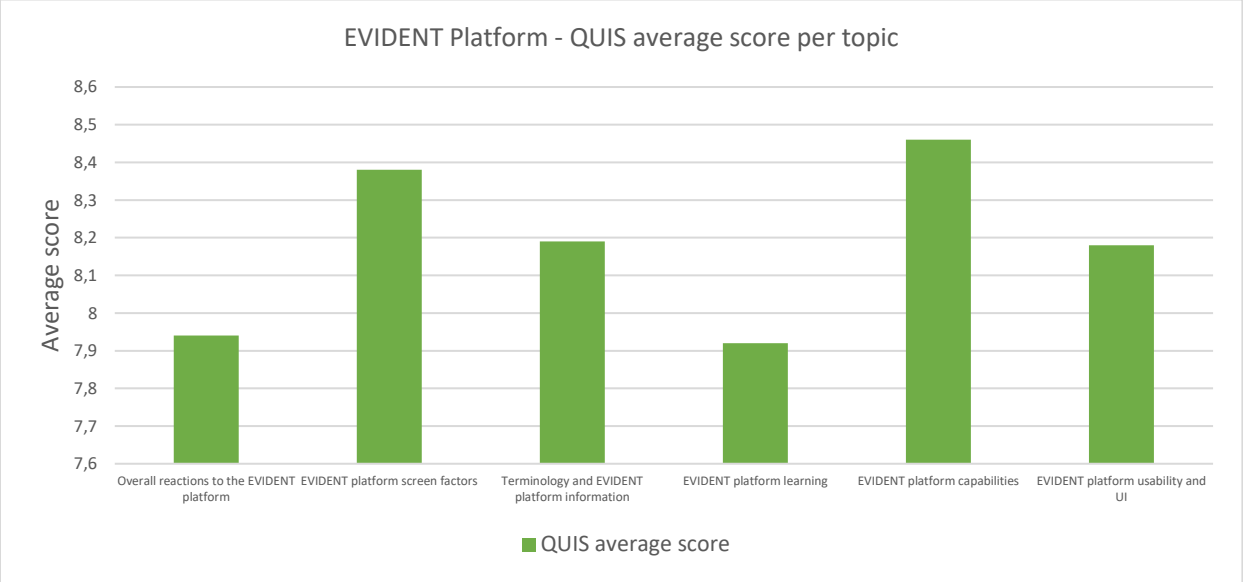


Figure 5: QUIS average scores per topic

5. EVIDENT platform training materials

The EVIDENT platform aims to facilitate the design and implementation of e-lab experiments for researchers, research institutions, and other stakeholders. For this reason, a lot of effort has been devoted by the consortium to creating a useful and helpful training guide for the EVIDENT platform and the provided services. This guide is hosted as a separate page on the EVIDENT platform and accompanies the users in every step of the implementation of e-lab experiments.

This guide describes in detail the following aspects:

- How the EVIDENT platform is being organized.
- How can someone create a new study (e-lab experiment).
- How to create a new questionnaire.
- How to create and link a new game.
- How a game can receive input from the EVIDENT platform.
- How a game can send output from the EVIDENT platform.
- How to support other devices such as smartphones.

The rest of the section presents the training materials prepared for the EVIDENT platform users[1]. The training guide is written in singular.

5.1 How the EVIDENT platform is organized

The EVIDENT platform constitutes a solution for designing and implementing online lab experiments (also referred to as a "study") by combining questionnaires and games. The platform allows the organiser of a study to combine questionnaires and games (also called "applications") in any order and use the output of one as an input to another.

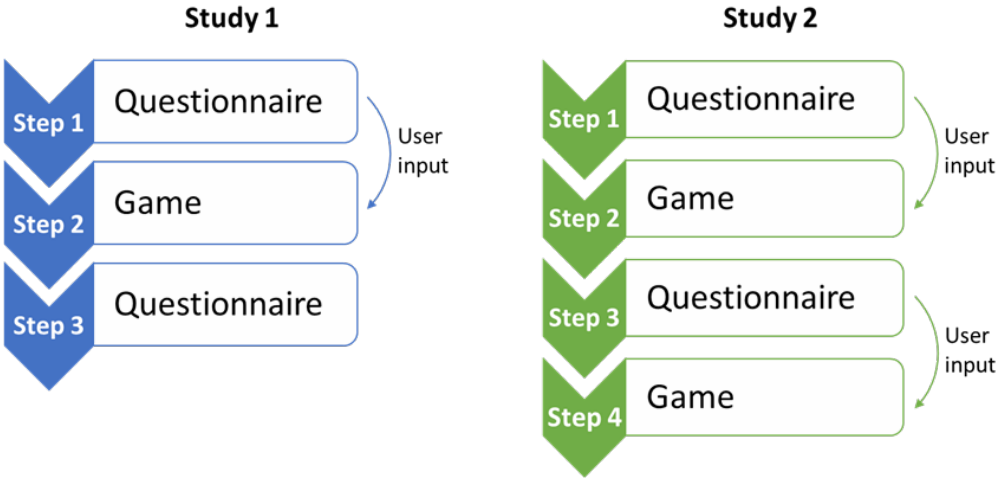


Figure 6: Example of two different e-lab experiments

For example, in Figure 6 above, we can see two different studies, study 1 and study 2, with different designs and configurations. In the first example, the study consists of three steps (a Questionnaire, a Game and a Questionnaire) where the game (step 2) receives input from the first questionnaire (step 1). In

[1] The training materials are hosted in the following address, but the user should be registered to access the corresponding page: https://platform.evident-h2020.eu/how-to-use

simple terms that means that the first questionnaire might ask participants' age and later the game uses this information to provide personalized gameplay. The second example consists of 4 steps (a Questionnaire, a Game, a Questionnaire and a Game), where each game receives input from the questionnaires in the previous steps. In this example, one could also use the replies in the first questionnaire as input to the second game.

After completing a study, the organizer can download the collected replies along with a rich set of demographic data collected from the responders while registering the platform.

5.2 Glossary

Table 3. Glossary for the EVIDENT platform

Term	Description
Study	The term study refers to an online lab experiment
Application	An application refers to a discrete step of a study. The EVIDENT platform supports two different types of applications, questionnaires and games.

5.3 Creating a study

To design and implement a lab experiment, the organizer needs to think about the research questions to be answered. The next step is to select the appropriate applications and start designing the study on the EVIDENT platform. After completing the registration, the organizer has access to the EVIDENT platform toolset through the following pages:

- "My Surveys"
- "My Games"

5.3.1 Creating a questionnaire

To create a new questionnaire, the organiser should select "My Surveys". Any previous questionnaires and the option to create a new one are listed on this page. The questionnaire designer is available through the "Create new" option. The questionnaire design page enables dragging and dropping any input field from the toolbox.

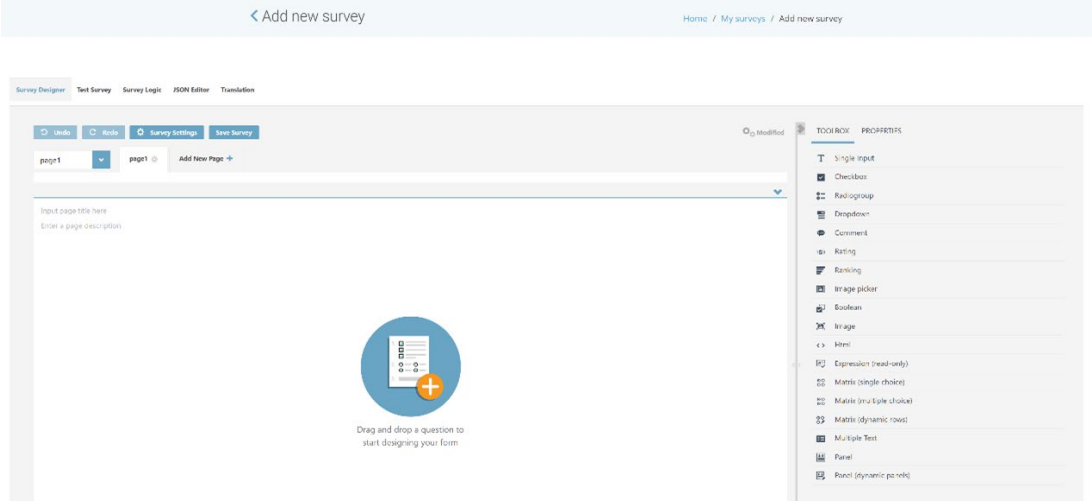


Figure 7: EVIDENT platform, questionnaire editor

Additionally, by using the 'Get your phone ready' option, the platform will automatically adjust it for mobile browsers. The idea of this page is to ask the participants to download the application from the corresponding application store.

5.3.2 Creating a game

The EVIDENT platform supports games created using the Unity graphics engine in the form of WebGL, Android, and iOS applications.

When designing a game, there are some aspects to consider such as:

- Do the games need to receive user input from previous steps?
- Are there other fields in the game that should be pre-configured?
- Which devices are supported (web/smartphones)?

In view of the EVIDENT project's needs, four user inputs (i.e., age, income, gender, and household members) were configured. The platform sends these data to the EVIDENT game while starting. Also, some game configuration fields were pre-configured before publishing the study and creating three identical games for three different versions (web/Android/iOS).

The EVIDENT platform adjusts on mobile screens; however, some smartphones cannot run games made as WebGL applications directly on their browsers. As a result, the organiser may need to create additional native smartphone applications. To this end, the EVIDENT platform offers APIs for a collection of smartphone applications.

Figure 8 below shows how the EVIDENT platform exchanges information with the different kinds of games (web/mobile applications).

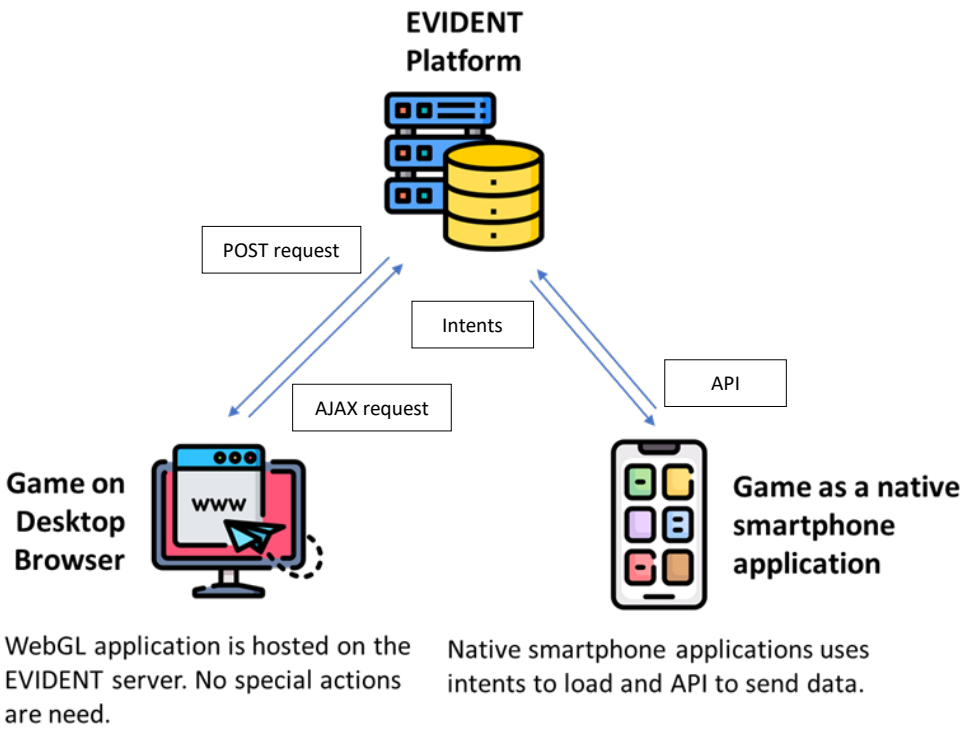


Figure 8: Methods used for information exchange between the EVIDENT platform and different kind of game versions

To upload a new game, the organiser should visit "My Games" page and then select "Create new". In the "new game" page, there are several required fields. The first two fields refer to the **title** and the game **description**. The next step is the **game file** which is the WebGL version of the game. More details on exporting the game in WebGL format are outlined in Section 5.3.2.4. **Game translation** is the translation JSON of the game in case the game is translated into multiple languages; more details on this are provided in section 5.3.2.1. The fields **android host**, **android scheme**, **android app package** and **IOS URL scheme** refer to the information needed for providing the game as a standalone Android and iOS application. Section 5.3.3 provides additional information on the creation of a mobile application.

The next part of the form is the **configurable fields**, which are the fields the game receives from the platform and uses dynamically. For example, the organiser can ask the name of the participant and use this information to greet the player at the beginning of the game. Finally, through the **game configurations**, the organiser can define some additional parameters. For example, in the EVIDENT serious game, a parameter indicating how often a household appliance may break down was defined.

5.3.2.1 Creating a WebGL version

In the WebGL version, the game receives input from the platform by using unityInstance.SendMessage function. The platform will use the SendMessage function twice to send the user input and the game configuration and the second time to send the translations.

The SendMessage function always sends a JSON field. Bellow, there are two examples, respectively for the configurable fields and game configuration, and for the translations.

Configurable fields and game configuration

```
{
        "section1": {
                "conf_field_1": 30,
                "conf_field_2": "Jose",
        },
        "section2": {
                "appliance_breakdown": true,
                "frequency": "120",
        }
}
```

Code block 2: Example of configurable fields and game configuration

Section 1 refers to configurable fields, while section 2 refers to the game configuration. The parameter names (e.g., conf_field_1, appliance_breakdown, etc.) are provided by the organiser and should be defined in the new game page. Moreover, Code block 3 provides a JSON file example of how the game handles the translation of messages into multiple languages.

Configurable fields and game configuration

```
{
        {
                "ID": 1,
                "Entry": "Hello..",
                "Eng": "Hello..",
                "Sw": "Hallå..",
                "Gr": "Καλησπέρα.."
        }
}
```

Code block 3: Example of game translations

5.3.2.2 Receiving input from the platform

In the starting scene of the game, the organiser should add the following two functions for receiving input from the platform.

Receive input from browser

```
public void TakeScenarioJSON(string json) {
   SharedSavedJsonData.answers = JsonConvert.DeserializeObject(json);
}

public void TakeConfigurationJSON(string json) {
   SharedSavedJsonData.configuration = JsonConvert.DeserializeObject(json);
StartCoroutine(changeLanguage());
}
```

Code block 4: C# code to receive input from the EVIDENT platform to the WebGL version of the game

Function TakeScenarioJSON() will be used to receive user input and game configuration while TakeConfigurationJSON() function will be used to parse the translations and call a coroutine to change the game language.

5.3.2.3 Sending output to the platform

After the game is complete, the organiser should send the collected data to the platform. To do so, the platform already implements a read function. To call this function, the code below should be added inside the project as MyPlugin.jlib.

Add the following code in MyPlugin.jlib

```
var MyPlugin = {
   ParseUserAnswers: function(str,done)
   {
   var answer = Pointer_stringify(str);
   var done = Pointer_stringify(done);
   var result = JSON.stringify({ answer:answer, done: done }) ;
   window.exportVariable(result);
   }
};

mergeInto(LibraryManager.library, MyPlugin);
```

Code block 5: C# code to send output to the EVIDENT platform from the WebGL version

Finally, in the script that will return the user's reply, a .dll file should be loaded.

Add this dll file in the script that will return user's reply

```
DllImport("__Internal")]
private static extern void ParseUserAnswers(string str, string done);
```

Code block 6: Example of DLL to send output to the EVIDENT platform

Finally, the ParseUserAnswers function can be used for sending the answers to the platform. The output should look like the following:

Make sure you set your reply in a form as the following

```
[{
   "answer_1": "option A",
   "answer_2": "60.000€",
}]
```

Code block 7: Example of output of the WebGL version of the game

The **done parameter** is used for sending the answer before the final answer of the user. When the **done parameter** is returned as true, the game will end. This is useful in case the organiser wants to return multiple answers from the game in different phases.

5.3.2.4 Exporting the game into WebGL format

The game can be easily exported as a WebGL game by selecting the webGL platform in build settings, as shown in Figure 9. This will create the following directories in the main project folder:

1. Build
2. StreamingAssets
3. TemplateData

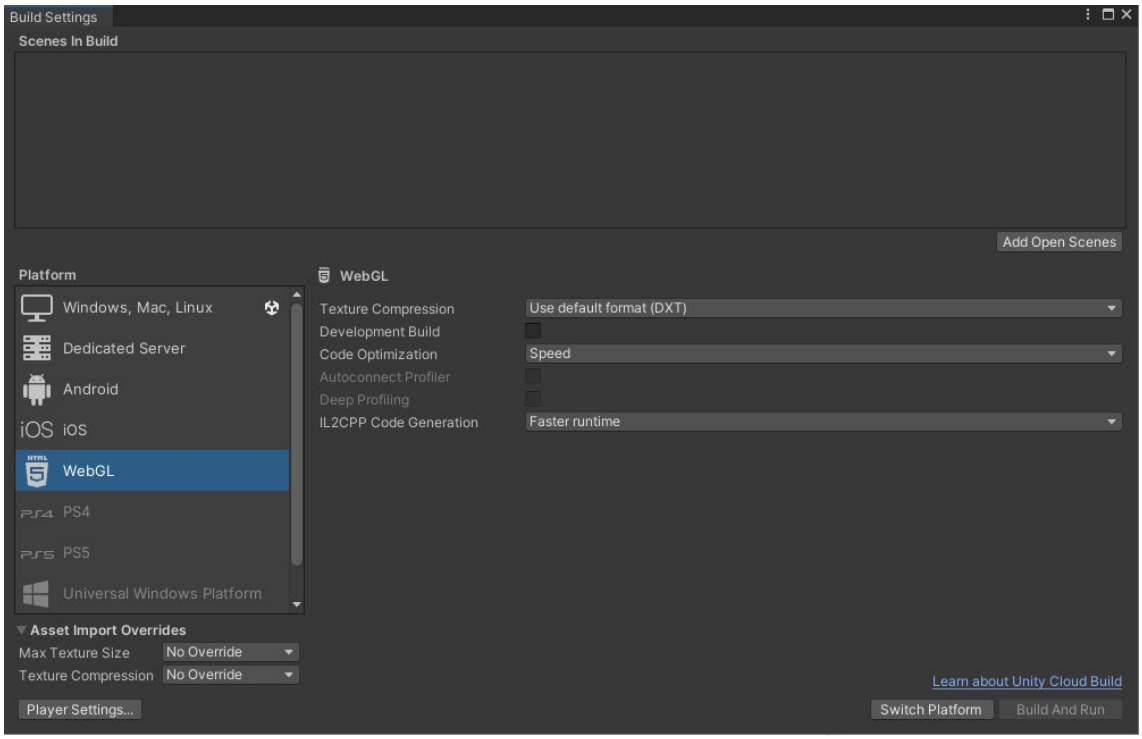


Figure 9: Export your game into WebGL format

5.3.3 Creating a Mobile version

The communication between the EVIDENT platform and the mobile application is achieved through an API. Consequently, some information is required to make the application successfully send data to the platform.

In both Android and IOS versions, the user has to receive a unique code from the application, when the application starts. With that unique code, the application returns the user's answers to the platform when the game ends. Moreover, the application receives the configuration file as a second input.

When the platform identifies a mobile browser, it uses intents to open the application. The way it does it is as follows:

Use Android intent

```
const url = "intent://ANDROID_HOST/#Intent;
scheme=ANDROID_SCHEME;
package=ANDROID_APP_PACKAGE;
end";
```

Code block 8: Example of using Android intent from JavaScript

Use iOS intent

```
const url = "IOS_URL_SCHEME://inputstring?end";
```

Code block 9: Example of using iOS intent from JavaScript

5.3.3.1 Android Host, Scheme and App Package

To enable a mobile browser to open an Android application, several options have to be defined in the manifest file. An example of aa manifest file is shown below:

Android app manifest file example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          xmlns:tools="http://schemas.android.com/tools">
 <application>
  <activity android:name="ANDROID_APP_PACKAGE"
          android:theme="@style/UnityThemeSelector" >
   <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
   <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="ANDROID_SCHEME" android:host="ANDROID_HOST" />
   </intent-filter>
  </activity>
 </application>
</manifest>
```

Code block 10: Example of android app manifest file

In the example above, the android host is equivalent to "ANDROID_HOST", the android scheme to "ANDROID_SCHEME" and the android app package to "ANDROID_APP_PACKAGE".

5.3.3.2 iOS URL Schema

URL schema is used as an identifier for launching applications and performing a set of commands on iOS devices. The schema name of a URL is the first part of a URL. An example of the info.plist file is listed below:

You can add the url scheme in info.plist

```
<key>CFBundleURLTypes</key>
<array>
  <dict>
    <key>CFBundleURLName</key>
    <string>com.ORGANISATION_NAME</string>
    <key>CFBundleURLSchemes</key>
    <array>
      <string>IOS_URL_SCHEME</string>
    </array>
  </dict>
</array>
```

Code block 11: Example of info.plist in iOS

5.3.3.3 Receive input from the platform

A source code example for receiving the unique ID and the configuration JSON file is provided below:

Use this code to read intent input in Android and iOS

```
public string  AndroidReadFromStreamingAssets(string fileName){

    #if UNITY_ANDROID

      string path = Path.Combine (Application.streamingAssetsPath, fileName);
      var loadingRequest = UnityWebRequest.Get(path);
      loadingRequest.SendWebRequest();
      while (!loadingRequest.isDone && !loadingRequest.isNetworkError
                                        && !loadingRequest.isHttpError);
      string result = System.Text.Encoding.UTF8.GetString(loadingRequest
                  .downloadHandler.data);
      return result;
    #elif UNITY_IOS
      return LoadFileFromStreamingAssetsIOS(fileName);
    #endif
    return null;
  }
  string LoadFileFromStreamingAssetsIOS(string fileName)
  {

    string path = Path.Combine(Application.streamingAssetsPath, fileName);

    // Handle iOS-specific file access restrictions
    path = "file://" + path;

    // Load the file using UnityWebRequest
    UnityWebRequest www = UnityWebRequest.Get(path);
    www.SendWebRequest();

    // Wait until the request is complete
    while (!www.isDone)
    {
    }
    // Extract the contents of the file as a string
    return www.downloadHandler.text;
  }

  private bool getIntentData () {
    #if UNITY_ANDROID
      return CreatePushClass (new AndroidJavaClass ("com.unity3d.player
                  .UnityPlayer"));
    #elif UNITY_IOS
      if (Application.absoluteURL.StartsWith("evidentseriousgame://")) {
        string uniqueID = "";
        string inputJson = "";
        string[] urlParts = Application.absoluteURL.Split('?');
        if (urlParts.Length > 1) {
          uniqueID = urlParts[1];
          if (urlParts.Length > 2) {
            inputJson = UnityWebRequest.UnEscapeURL(urlParts[2]);
          }
        }

        string malfunctionScenarios = AndroidReadFromStreamingAssets
                                              ("Serious_Game_Malfunction_Scenarios.json");
        TakeScenarioJSON(malfunctionScenarios);
        if(uniqueID!=null&&!uniqueID.Equals("")){ // iOS from platform
          TakeConfigurationJSON(inputJson);
```

```
            SharedSavedJsonsData.uniqueId = uniqueID;
                    } else { // iOS with wrond id(SomethingWentWrong
                        string configuration = AndroidReadFromStreamingAssets
                                ("Serious_Game_Configuration.json");
            TakeConfigurationJSON(configuration);
          }
        }else{ // iOS Standalone(not from platform
          selectExperiment.SetActive(false);
          string configuration = configuration = AndroidReadFromStreamingAssets
                                                ("Serious_Game_Configuration.json");
          TakeConfigurationJSON(configuration);
          string malfunctionScenarios = AndroidReadFromStreamingAssets
                                                ("Serious_Game_Malfunction_Scenarios.json");
          TakeScenarioJSON(malfunctionScenarios);
          return true;
        }
    #endif
    return false;
}


public bool CreatePushClass (AndroidJavaClass UnityPlayer) {
    #if (UNITY_IOS || UNITY_ANDROID)
        AndroidJavaObject currentActivity = UnityPlayer
                    .GetStatic ("currentActivity");
        AndroidJavaObject intent = currentActivity.Call ("getIntent");
        AndroidJavaObject extras = GetExtras (intent);

        if (extras != null) {
          string uniqueID = GetProperty (extras, "param1");
          string configuration = GetProperty (extras, "param2");
          string malfunctionScenarios = AndroidReadFromStreamingAssets
                                                ("Serious_Game_Malfunction_Scenarios.json");
          TakeScenarioJSON(malfunctionScenarios);
          if(uniqueID!=null&&!uniqueID.Equals("")){ // Android from platform
            selectExperiment.SetActive(false);
            SharedSavedJsonsData.uniqueId = uniqueID;
          }else{ // Android with no id(Something went wrong)
            configuration = AndroidReadFromStreamingAssets
                                                    ("Serious_Game_Configuration.json");
            TakeConfigurationJSON(configuration);
          }
          return true;
        }else{ // Android Standalone(not from platform)
          string malfunctionScenatios = AndroidReadFromStreamingAssets
                                                ("Serious_Game_Malfunction_Scenarios.json");
          TakeScenarioJSON(malfunctionScenatios);
          string configuration = AndroidReadFromStreamingAssets
                                                ("Serious_Game_Configuration.json");
          TakeConfigurationJSON(configuration);
          return true;
        }
    #endif
    return false;
}

private AndroidJavaObject GetExtras (AndroidJavaObject intent) {
    AndroidJavaObject extras = null;
```

```
        try {
            extras = intent.Call ("getExtras");
        } catch (Exception e) {
            Debug.Log (e.Message);
        }

        return extras;
    }

    private string GetProperty (AndroidJavaObject extras, string name) {
        string s = string.Empty;

        try {
            s = extras.Call ("getString", name);
        } catch (Exception e) {
            Debug.Log (e.Message);
        }

        return s;
    }
}
```

Code block 12: C# code to read intent input in Android and iOS

5.3.3.4 Send output to the platform

After the game is completed, the organiser sends the collected data to the platform through an endpoint. To reach the correct endpoint, the unique ID from the input of the serious game in Android and IOS is needed. The endpoint has the following form:

EVIDENT platform endpoint to send data from mobile applications
https://platform.evident-h2020.eu/api/session_replies/"+uniqueId+"/"+flag

Code block 13: EVIDENT platform REST API endpoint to receive data from mobile applications

The parameter flag is used in the same way as in the web version. In this respect, the parameter should be set to True upon game completion.

5.3.3.5 Export your game into a mobile format

The respective processes for exporting the game into Android or iOS version can be found through the following links:

- Android - Google Play: Video
- iOS - App Store: Video

5.3.4 Wrapping everything up

In this step, the goal is to create a study combining the questionnaire and the game. To do so, the organiser should visit the "My Sessions" page and select the "Create new" option. Also, the following fields have to be filled in:

- **Password:** This option enables limiting the study only to specific participants.
- **Show only to anonymous users:** This option refers to anonymous participation. If checked, anyone (registered or unregistered user) can participate in the study. If not checked, only registered users can participate.
- **Debug:** If checked, this option will print the user input to the game page.

- **Proof of work:** If checked, each participant will receive a unique participation code that can later be used in services such as Prolific, Amazon MTrunk, Microworkers, etc.

The last step in this process is to select the applications that are linked to this study. To do this, the organiser should select the type of application through the "Add a new application" option. Finally, an example of adding a new session form is showcased in Figure 10.

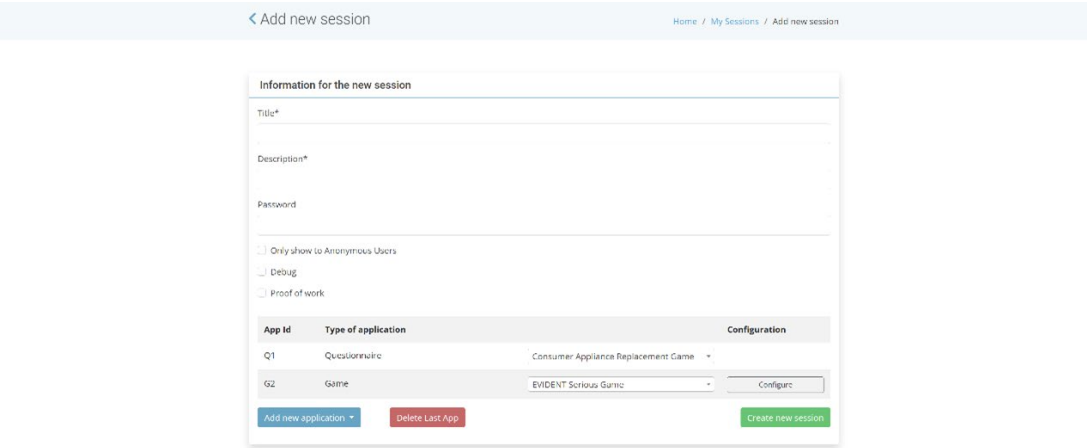


Figure 10: EVIDENT platform, add a new session form

5.4 Collecting data

Each study is linked to a unique URL for facilitating the sharing of the study with the participants. The EVIDENT platform supports three different file formats, namely Excel, CSV, and JSON, for accumulating and exporting the data collected from the participants.

6. Performance and Scalability Assessment

Through the use of GitHub activities, this section will demonstrate the EVIDENT platform's continuous integration and deployment process. Additionally, the methods employed by the EVIDENT project to publicize the platform and draw people to it are described.

6.1 Continues Integration / Continues Development (CI/CD)

6.1.1 Theoretical Foundations

The ideas of Continuous Development (CD) and Continuous Integration (CI) serve as the basis of theory in the dynamic world of software development. These guiding principles create a strong basis for the ongoing growth of platforms and are firmly ingrained in the Agile and DevOps philosophies.

We examine the theoretical foundations of CD and CI in this section:

1. **Agile Principles:** Agile ideals like client participation, flexibility in the face of change, and the regular release of usable software are strongly matched with CD and CI. These guidelines emphasize the necessity for platforms to adapt to the constantly changing needs of their customers.
2. **DevOps Culture:** The DevOps culture is closely related to the core of CI/CD. It encourages a coordinated strategy while removing boundaries between the development and operations teams. This culture shift puts automation, quick feedback loops, and a relentless dedication to continuous improvement first.
3. **Automation:** The automation of essential development and deployment procedures is a basic theoretical component of CD/CI. Automation reduces the need for manual intervention, speeds up development cycles, and maintains consistency and dependability throughout the platform's development.

6.1.2 Practical Significance in Platform Development

CD and CI are of the greatest importance in the field of platform development for a number of reasons:

a. **Rapid Iteration:** Platforms must quickly iterate in order to adapt to changing user needs and market expectations. Quick feature development and deployment, made possible by CD, help keep the platform competitive and up-to-date.
b. **Quality Assurance:** To ensure the stability and dependability of the platform, CI implements strict testing and validation procedures. For platforms where unavailability or data loss might have serious repercussions, this is especially important.
c. **Risk Mitigation:** CD/CI minimizes the risk associated with manual interventions, lowering the possibility of mistakes and security vulnerabilities, by automating repetitive operations and utilising version control.
d. **Enhanced Collaboration:** These procedures encourage communication between the teams in charge of development, testing, and operations. Cross-functional teams are encouraged to collaborate by the constant feedback loop, which produces higher-quality results.
e. **Scalability:** CD/CI frameworks expand along with platforms. Even when the platform's complexity rises, they make sure that the development process stays effective.
f. **Competitive Advantage:** Adopting CD/CI can set you apart from the competition. Platforms with quicker and higher-quality feature delivery have an advantage in luring and keeping consumers.

As a result, platform development is driven by the fundamental ideas of continuous development and continuous integration, which are not merely practical techniques but also have a strong theoretical foundation. Platform teams may successfully traverse the shifting technological landscape and provide long-lasting value to their users by adopting these principles.

6.2 Implementing CI/CD with GitHub Actions

This section covers the essential steps to establish a dependable CI/CD pipeline for a platform utilising GitHub Actions.

6.2.1 Setting up GitHub Actions

First of all, in the GitHub repository, there is an “Action” tab. GitHub Actions uses YAML configuration files to define workflows. Create a .github/workflows directory in the repository and add a YAML configuration file named deploy.yml, as depicted in Figure 11.

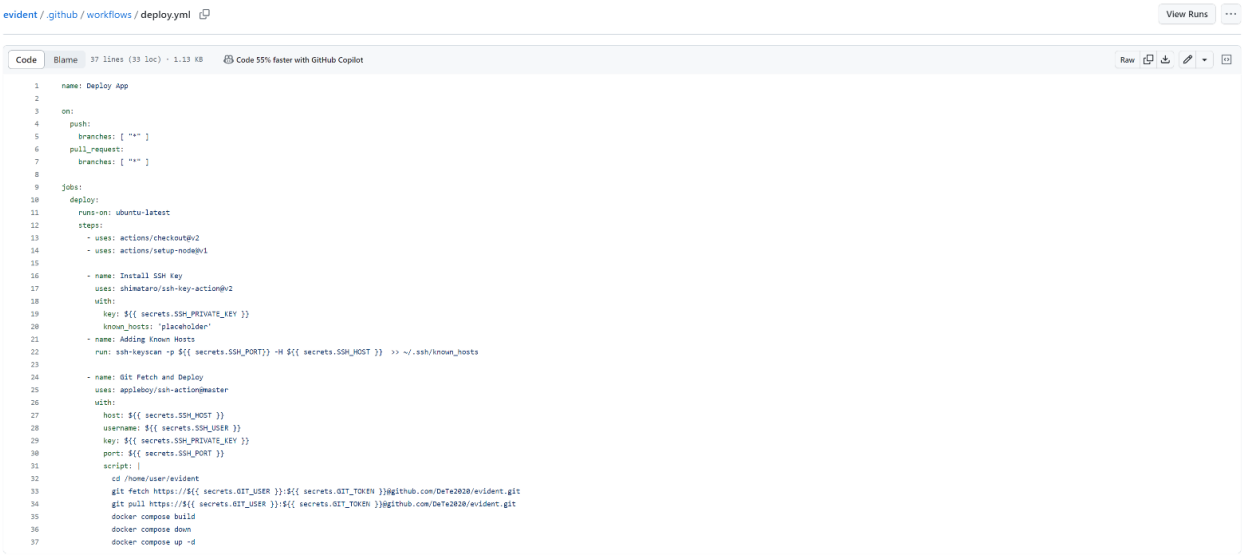


Figure 11: Github actions tab

This YAML configuration file sets the stage for a CI/CD pipeline by defining the workflow structure, specifying the trigger (pushes to the main branch), and laying out the necessary steps for building, testing, and deploying the platform.

6.2.2 Continuous Integration

The build and test phases are required to be customised to the unique requirements of the platform in order for the CI/CD pipeline to genuinely function properly. The default “npm” commands are replaced with the actual commands necessary for the project. These commands are stored in another YAML file, the docker-image.yml. In this file the docker image is built, to use it later on the deploy.yml file. The code of the docker-image.yml is displayed in the code block below and described later in this section.

Docker-image.yml

```
evident/.github/workflows/docker-image.yml

name: Docker Image CI

on:

        push:
        branches: [ "master" ]
        pull_request:
        branches: [ "master" ]

jobs:

        build:

                runs-on: ubuntu-latest

                steps:
                - uses: actions/checkout@v3
                - name: Build the Docker image
                run: docker build . --file Dockerfile --tag evident:$(date +%s)
```

Code block 14: Code of Docker-image.yml in yaml

The "Docker Image Build" workflow is designed to automate the CI process of building Docker images whenever changes are made to your code. It triggers both pushes and pull requests to the "master" branch, ensuring that Docker images are continuously integrated and updated. The first step, "Checkout Code," utilizes the actions/checkout@v3 action to fetch the latest code from your GitHub repository. This step ensures that the workflow operates on the up-to-date codebase. The procedure then moves on to the "Build Docker Image" phase after code checkout. Here, a Docker image is generated from the code using the docker build command. The template for building the image is the given Docker file. This ensures that the platform was built correctly and passed all tests as part of the CI/CD pipeline.

6.2.3 Continuous Deployment

The "Deployment" workflow automates the deployment of your application to a remote server whenever changes are pushed or pull requests are submitted to any branch. The deploy.yml is displayed in the code block below and described later in this section.

Deploy.yml

```
name: Deploy App

on:
        push:
                branches: [ "*" ]
        pull_request:
                branches: [ "*" ]

jobs:
        deploy:
                runs-on: ubuntu-latest
                steps:
                        - uses: actions/checkout@v2
                        - uses: actions/setup-node@v1

                        - name: Install SSH Key
                        uses: shimataro/ssh-key-action@v2
                        with:
                                key: ${{ secrets.SSH_PRIVATE_KEY }}
                                known_hosts: 'placeholder'
                        - name: Adding Known Hosts
                        run: ssh-keyscan -p ${{ secrets. SSH_PORT}} -H ${{ secrets.SSH_HOST }} >> ~/.ssh/known_hosts

                        - name: Git Fetch and Deploy
                        uses: appleboy/ssh-action@master
                        with:
                                host: ${{ secrets. SSH_HOST}}
                                username: ${{ secrets.SSH_USER }}
                                key: ${{ secrets. SSH_PRIVATE_KEY }}
                                port: ${{ secrets. SSH_PORT}}
                                script: |
                                        cd /home/user/evident
                                        git fetch https://${{ secrets. GIT_USER }}:${{ secrets. GIT_TOKEN
}}@github.com/DeTe2020/evident.git
                                        git pull https://${{ secrets. GIT_USER }}:${{ secrets.GIT_TOKEN
}}@github.com/DeTe2020/evident.git
                                        docker compose build
                                        docker compose down
                                        docker compose up -d
```

Code block 15: Code of Deploy.yml in yaml

Similar to the CI workflow, the initial step is to check out the latest code using the "**actions/checkout@v2**" action. Following code checkout, the workflow sets up Node.js using the "**actions/setup-node@v1**" action, enabling the execution of JavaScript-based tasks in your deployment process. To establish a secure connection with the remote server, the workflow installs an SSH private key, stored as a secret in a different YAML, named secred.yml. This key is essential for secure authentication during deployment. Finally, the workflow connects to the remote server via SSH and executes commands, for fetching the latest code from the GitHub repository, building Docker images, and deploying the application in a Docker container. This ensures continuous deployment whenever there are code changes.

6.3 Recommendation and Future Work

The consortium foresees action plans for effective communication with end-users and relevant stakeholders, in order to increase the visibility and usability of the platform. Usability will be increased through user participation since the accumulation of user feedback will positively drive future platform

updates. In terms of communication activities, the consortium has already taken some measures to maximise stakeholder engagement. Such measures are summarised in Table 4.

Table 4. Actions about user engagement.

Actions Until M34	Action description
Website updates	The consortium has updated the EVIDENT website to include a description and a link to the platform (https://evident-h2020.eu/evidentplatform/), as illustrated in Figure 12 and Figure 13 below. Also, a separate page is created that provides access to the serious game of the project (https://evident-h2020.eu/seriousgame/). In this way, more users have the opportunity to come across and join the EVIDENT ecosystem.
Blog posts	The consortium has released a number of blog posts related to the platform: • 1 post regarding research data availability (https://evident-h2020.eu/evident-research-data-results-available-to-all/) • 1 post regarding the design and architecture of the platform (https://evident-h2020.eu/unlocking-the-value-of-evident-platform-architecture-utilities-and-benefits-exploration/). • 1 post about data privacy and anonymisation (https://evident-h2020.eu/balancing-energy-efficiency-and-privacy-protection-leveraging-big-data-and-anonymization-techniques/)
Social media posts	The consortium has produced 3 social media posts, that advertise the blog articles related to the platform.

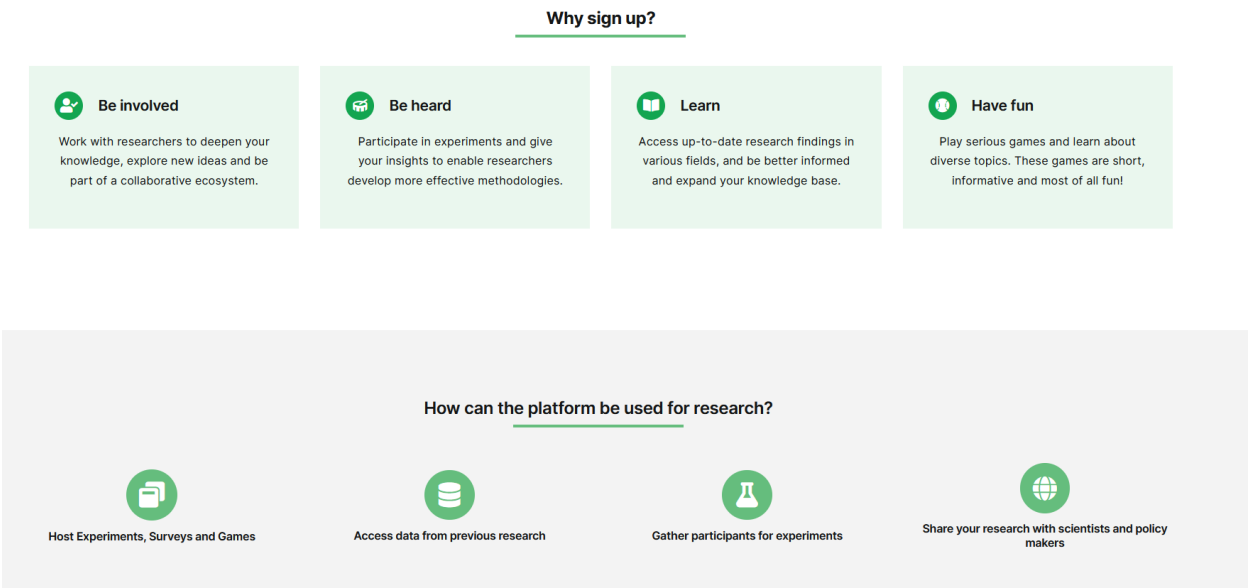


Figure 12. Description of the platform, taken from the EVIDENT's website.

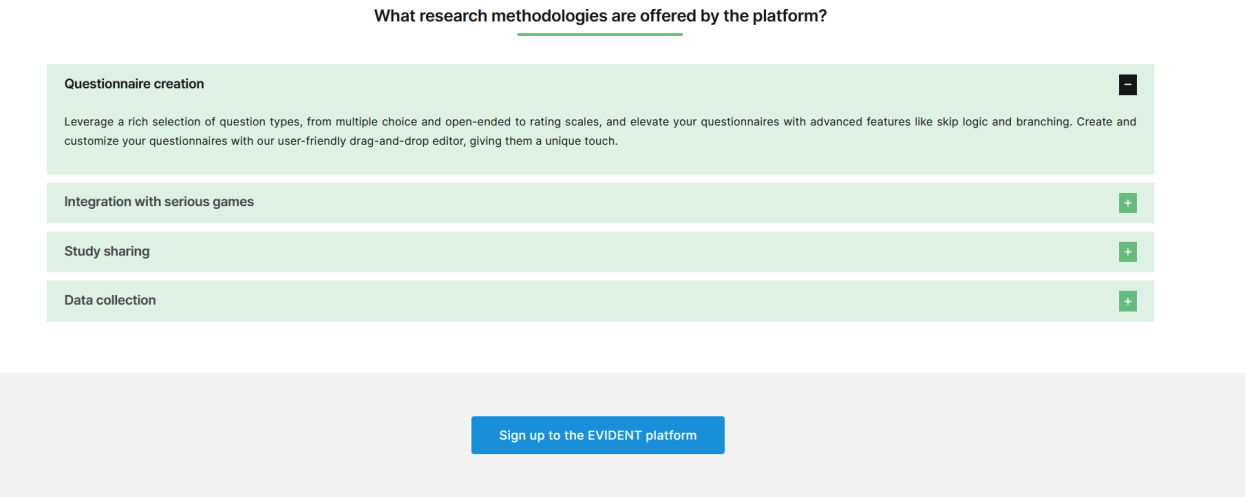


Figure 13. Frequently asked questions about the platform, taken from the EVIDENT's website.

The consortium has also made plans for future communication activities that target to increase stakeholder engagement with the platform. Such plans are expected to last, at least 1 more year after the project ends, and are depicted in Table 5.

Table 5.Future actions about user engagement.

Future actions (until at least 1 year after the project ends)	Action description
Blog posts	2 Blog posts related to the platform: • 1 blog post that will function as a tutorial for new users • 1 blog post that further advertises the platform's features and benefits.
Social media posts	>5 posts on the platform's features and advantages at the consortium's main communication channel X (former Twitter). The posts will be tailored to the public. Currently, EVIDENT maintains an active X account with more than 1060 followers and thus, a significant user flow is expected to stem from the project's social media profile.

7. Conclusion

In conclusion, in this document, the verification and validation progress of the EVIDENT platform is described. Starting with the theoretical framework of these terms and exploiting the provided methodologies and best practices, the operational framework of the validation and verification process of the EVIDENT platform is exploited. By providing the usage of unit tests, the deliverable outlines the part that unit testing plays in the platform's quality assurance process, highlighting how important it is for finding and fixing errors and inconsistencies. The fundamental principles that guide the development of successful unit tests are provided in detail, ensuring that the testing procedure perfectly synchronizes with the platform's objectives of precision and accuracy. Moreover, a part of the deliverable also serves as a guide for the users to conduct research through the platform and, if they want, include their own serious games in the research. Finally, the Evident consortium's strategies for disseminating the platform, attracting users to the platform, and improving the platform's usability by soliciting feedback from both technical and non-technical users are utilised.

8. References

[1] J. P. Chin, V. A. Diehl, and L. K. Norman, “Development of an instrument measuring user satisfaction of the human-computer interface,” SIGCHI Conference on Human factors in Computing Systems - CHI ’88. ACM Press, 1988.

[2] “Qualitrics,” [Online]. Available: https://www.qualtrics.com/.

[3] “Survey Monkey,” [Online]. Available: https://www.surveymonkey.com/.

[4] “Google Forms,” [Online]. Available: https://www.google.com/forms/about/.

[5] D. Rubio, “Django Design Principles,” Webforefront, [Online]. Available: https://www.webforefront.com/django/designprinciples.html.

9. Appendices

9.1 Appendix 1: EVIDENT Platform unit tests log

Topic: Log of unit tests

```
Creating test database for alias 'default' ('test_evident_postgres')...

test_description_label (experiments.tests.test_forms.AddSessionFormTest) ... ok
test_open_anonymous_label (experiments.tests.test_forms.AddSessionFormTest) ... ok
test_password_label (experiments.tests.test_forms.AddSessionFormTest) ... ok
test_title_label (experiments.tests.test_forms.AddSessionFormTest) ... ok
test_answers_label (experiments.tests.test_forms.AddSessionReplyFormTest) ... ok
test_Session_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_Session_name (experiments.tests.test_models.SessionRepliesTest) ... ok
test_answers_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_answers_value (experiments.tests.test_models.SessionRepliesTest) ... ok
test_created_at_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_current_application_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_current_application_type_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_current_application_type_value (experiments.tests.test_models.SessionRepliesTest) ... ok
test_current_application_value (experiments.tests.test_models.SessionRepliesTest) ... ok
test_object_format (experiments.tests.test_models.SessionRepliesTest) ... ok
test_updated_at_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_user_label (experiments.tests.test_models.SessionRepliesTest) ... ok
test_user_name (experiments.tests.test_models.SessionRepliesTest) ... ok
test_description_label (experiments.tests.test_models.SessionTest) ... ok
test_description_name (experiments.tests.test_models.SessionTest) ... ok
test_object_format (experiments.tests.test_models.SessionTest) ... ok
test_password_label (experiments.tests.test_models.SessionTest) ... ok
test_password_name (experiments.tests.test_models.SessionTest) ... ok
test_published_label (experiments.tests.test_models.SessionTest) ... ok
test_published_value (experiments.tests.test_models.SessionTest) ... ok
test_title_label (experiments.tests.test_models.SessionTest) ... ok
test_title_name (experiments.tests.test_models.SessionTest) ... ok
test_total_answers_label (experiments.tests.test_models.SessionTest) ... ok
test_total_answers_value (experiments.tests.test_models.SessionTest) ... ok
test_unique_id_label (experiments.tests.test_models.SessionTest) ... ok
test_user_label (experiments.tests.test_models.SessionTest) ... ok
test_user_name (experiments.tests.test_models.SessionTest) ... ok
test_add_session_context (experiments.tests.test_views.TestViews) ... ok
test_add_session_denied_if_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_add_session_get_request (experiments.tests.test_views.TestViews) ... ok
test_add_session_logged_in (experiments.tests.test_views.TestViews) ... ok
test_add_session_post_request (experiments.tests.test_views.TestViews) ... ok
test_add_session_template_used (experiments.tests.test_views.TestViews) ... ok
test_delete_session_context (experiments.tests.test_views.TestViews) ... ok
test_delete_session_error (experiments.tests.test_views.TestViews) ... ok
test_delete_session_redirect_when_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_delete_session_url (experiments.tests.test_views.TestViews) ... ok
test_edit_session_context (experiments.tests.test_views.TestViews) ... ok
test_edit_session_denied_if_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_edit_session_get_request (experiments.tests.test_views.TestViews) ... ok
```

```
test_edit_session_logged_in (experiments.tests.test_views.TestViews) ... ok
test_edit_session_post_request (experiments.tests.test_views.TestViews) ... ok
test_edit_session_template_used (experiments.tests.test_views.TestViews) ... ok
test_experiments_allowed_if_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_experiments_context (experiments.tests.test_views.TestViews) ... ok
test_experiments_get_request (experiments.tests.test_views.TestViews) ... ok
test_experiments_logged_in (experiments.tests.test_views.TestViews) ... ok
test_experiments_template_used (experiments.tests.test_views.TestViews) ... ok
test_get_session_context (experiments.tests.test_views.TestViews) ... ok
test_get_session_url (experiments.tests.test_views.TestViews) ... ok
test_logged_in_uses_correct_template (experiments.tests.test_views.TestViews) ... ok
test_login (experiments.tests.test_views.TestViews) ... ok
test_my_experiments_context (experiments.tests.test_views.TestViews) ... ok
test_my_experiments_denied_if_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_my_experiments_get_request (experiments.tests.test_views.TestViews) ... ok
test_my_experiments_logged_in (experiments.tests.test_views.TestViews) ... ok
test_my_experiments_template_used (experiments.tests.test_views.TestViews) ... ok
test_participate_session_context (experiments.tests.test_views.TestViews) ... ok
test_participate_session_context_if_0 (experiments.tests.test_views.TestViews) ... ok
test_participate_session_get_request (experiments.tests.test_views.TestViews) ... ok
test_participate_session_if_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_participate_session_logged_in (experiments.tests.test_views.TestViews) ... ok
test_participate_session_template_used (experiments.tests.test_views.TestViews) ... ok
test_participate_session_template_used_if_0 (experiments.tests.test_views.TestViews) ... ok
test_password_verify_context_on_password_verify (experiments.tests.test_views.TestViews) ... ok
test_password_verify_context_on_unverified_password (experiments.tests.test_views.TestViews) ... ok
test_password_verify_error (experiments.tests.test_views.TestViews) ... ok
test_password_verify_url (experiments.tests.test_views.TestViews) ... ok
test_publish_session_context (experiments.tests.test_views.TestViews) ... ok
test_publish_session_error (experiments.tests.test_views.TestViews) ... ok
test_publish_session_redirect_when_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_publish_session_url (experiments.tests.test_views.TestViews) ... ok
test_response_is_json_delete_session (experiments.tests.test_views.TestViews) ... ok
test_response_is_json_get_session (experiments.tests.test_views.TestViews) ... ok
test_response_is_json_password_verify (experiments.tests.test_views.TestViews) ... ok
test_response_is_json_publish_session (experiments.tests.test_views.TestViews) ... ok
test_response_is_json_unpublish_session (experiments.tests.test_views.TestViews) ... ok
test_unpublish_session_context (experiments.tests.test_views.TestViews) ... ok
test_unpublish_session_error (experiments.tests.test_views.TestViews) ... ok
test_unpublish_session_redirect_when_not_logged_in (experiments.tests.test_views.TestViews) ... ok
test_unpublish_session_url (experiments.tests.test_views.TestViews) ... ok
test_about (homepage.tests.TestViews) ... ok
test_about_context (homepage.tests.TestViews) ... ok
test_about_template_used (homepage.tests.TestViews) ... ok
test_change_language_context (homepage.tests.TestViews) ... ok
test_change_language_context_authenticated_user (homepage.tests.TestViews) ... ok
test_change_language_url (homepage.tests.TestViews) ... ok
test_community (homepage.tests.TestViews) ... ok
test_community_context (homepage.tests.TestViews) ... ok
test_community_template_used (homepage.tests.TestViews) ... ok
test_data_protection (homepage.tests.TestViews) ... ok
test_data_protection_context (homepage.tests.TestViews) ... ok
test_data_protection_template_used (homepage.tests.TestViews) ... ok
```

```
test_data_usage (homepage.tests.TestViews) ... ok
test_data_usage_context (homepage.tests.TestViews) ... ok
test_data_usage_template_used (homepage.tests.TestViews) ... ok
test_home (homepage.tests.TestViews) ... ok
test_home_context (homepage.tests.TestViews) ... ok
test_home_template_used (homepage.tests.TestViews) ... ok
test_logged_in_uses_correct_template (homepage.tests.TestViews) ... ok
test_login (homepage.tests.TestViews) ... ok
test_policymakers (homepage.tests.TestViews) ... ok
test_policymakers_context (homepage.tests.TestViews) ... ok
test_policymakers_template_used (homepage.tests.TestViews) ... ok
test_privacy_policy (homepage.tests.TestViews) ... ok
test_privacy_policy_context (homepage.tests.TestViews) ... ok
test_privacy_policy_template_used (homepage.tests.TestViews) ... ok
test_researchers (homepage.tests.TestViews) ... ok
test_researchers_context (homepage.tests.TestViews) ... ok
test_researchers_template_used (homepage.tests.TestViews) ... ok
test_response_is_json_change_language (homepage.tests.TestViews) ... ok
test_terms_and_policy (homepage.tests.TestViews) ... ok
test_terms_and_policy_context (homepage.tests.TestViews) ... ok
test_terms_and_policy_template_used (homepage.tests.TestViews) ... ok
test_date_label (newsletter.tests.TestNewsletter) ... ok
test_email_label (newsletter.tests.TestNewsletter) ... ok
test_email_name (newsletter.tests.TestNewsletter) ... ok
test_form_email_label (newsletter.tests.TestNewsletter) ... ok
test_news_label (newsletter.tests.TestNewsletter) ... ok
test_news_name (newsletter.tests.TestNewsletter) ... ok
test_newsletter_context (newsletter.tests.TestNewsletter) ... ok
test_newsletter_context_existing_user (newsletter.tests.TestNewsletter) ... ok
test_newsletter_url (newsletter.tests.TestNewsletter) ... ok
test_response_is_json_newsletter (newsletter.tests.TestNewsletter) ... ok
test_sessions_label (newsletter.tests.TestNewsletter) ... ok
test_sessions_list_label (newsletter.tests.TestNewsletter) ... ok
test_sessions_list_name (newsletter.tests.TestNewsletter) ... ok
test_sessions_name (newsletter.tests.TestNewsletter) ... ok
test_description_label (seriousgames.tests.test_forms.AddSeriousGameFormTest) ... ok
test_title_label (seriousgames.tests.test_forms.AddSeriousGameFormTest) ... ok
test_build_folder_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_configurable_fields_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_description_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_description_name (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_gameDirectory_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_gameDirectory_name (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_game_configuration_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_game_name_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_game_name_name (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_on_session_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_on_session_value (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_published_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_published_value (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_title_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_title_name (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_user_label (seriousgames.tests.test_models.SeriousGamesTest) ... ok
```

```
test_user_name (seriousgames.tests.test_models.SeriousGamesTest) ... ok
test_add_game_context (seriousgames.tests.test_views.TestViews) ... ok
test_add_game_get_request (seriousgames.tests.test_views.TestViews) ... ok
test_add_game_if_not_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_add_game_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_add_game_post_request (seriousgames.tests.test_views.TestViews) ... ok
test_add_game_template_used (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_context (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_get_request (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_if_not_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_post_request (seriousgames.tests.test_views.TestViews) ... ok
test_edit_game_template_used (seriousgames.tests.test_views.TestViews) ... ok
test_logged_in_uses_correct_template (seriousgames.tests.test_views.TestViews) ... ok
test_login (seriousgames.tests.test_views.TestViews) ... ok
test_my_serious_game_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_my_seriousgames_context (seriousgames.tests.test_views.TestViews) ... ok
test_my_seriousgames_get_request (seriousgames.tests.test_views.TestViews) ... ok
test_my_seriousgames_if_not_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_my_seriousgames_template_used (seriousgames.tests.test_views.TestViews) ... ok
test_serious_game_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_seriousgames_context (seriousgames.tests.test_views.TestViews) ... ok
test_seriousgames_get_request (seriousgames.tests.test_views.TestViews) ... ok
test_seriousgames_if_not_logged_in (seriousgames.tests.test_views.TestViews) ... ok
test_seriousgames_template_used (seriousgames.tests.test_views.TestViews) ... ok
test_add_survey_context (surveys.tests.TestSurveys) ... ok
test_add_survey_denied_if_not_logged_in (surveys.tests.TestSurveys) ... ok
test_add_survey_get_request (surveys.tests.TestSurveys) ... ok
test_add_survey_logged_in (surveys.tests.TestSurveys) ... ok
test_add_survey_template_used (surveys.tests.TestSurveys) ... ok
test_created_date_label (surveys.tests.TestSurveys) ... ok
test_description_label (surveys.tests.TestSurveys) ... ok
test_description_name (surveys.tests.TestSurveys) ... ok
test_edit_survey_context (surveys.tests.TestSurveys) ... ok
test_edit_survey_denied_if_not_logged_in (surveys.tests.TestSurveys) ... ok
test_edit_survey_get_request (surveys.tests.TestSurveys) ... ok
test_edit_survey_logged_in (surveys.tests.TestSurveys) ... ok
test_edit_survey_template_used (surveys.tests.TestSurveys) ... ok
test_object_format (surveys.tests.TestSurveys) ... ok
test_on_session_label (surveys.tests.TestSurveys) ... ok
test_on_session_value (surveys.tests.TestSurveys) ... ok
test_published_label (surveys.tests.TestSurveys) ... ok
test_published_value (surveys.tests.TestSurveys) ... ok
test_q_n_a_label (surveys.tests.TestSurveys) ... ok
test_q_n_a_value (surveys.tests.TestSurveys) ... ok
test_response_is_json_save_survey (surveys.tests.TestSurveys) ... ok
test_save_survey_context (surveys.tests.TestSurveys) ... ok
test_save_survey_context_if_fail (surveys.tests.TestSurveys) ... ok
test_save_survey_url (surveys.tests.TestSurveys) ... ok
test_surveys_context (surveys.tests.TestSurveys) ... ok
test_surveys_denied_if_not_logged_in (surveys.tests.TestSurveys) ... ok
test_surveys_get_request (surveys.tests.TestSurveys) ... ok
test_surveys_logged_in (surveys.tests.TestSurveys) ... ok
```

```
test_surveys_template_used (surveys.tests.TestSurveys) ... ok
test_title_label (surveys.tests.TestSurveys) ... ok
test_title_name (surveys.tests.TestSurveys) ... ok
test_user_label (surveys.tests.TestSurveys) ... ok
test_user_name (surveys.tests.TestSurveys) ... ok
test_user_organisation_justification_label (users.tests.test_forms.ApplyForOrganizationFormTest) ... ok
test_user_organisation_label (users.tests.test_forms.ApplyForOrganizationFormTest) ... ok
test_user_position_in_organisation_label (users.tests.test_forms.ApplyForOrganizationFormTest) ... ok
test_bio_label (users.tests.test_forms.ProfileUpdateFormTest) ... ok
test_image_label (users.tests.test_forms.ProfileUpdateFormTest) ... ok
test_language_label (users.tests.test_forms.ProfileUpdateFormTest) ... ok
test_timezone_label (users.tests.test_forms.ProfileUpdateFormTest) ... ok
test_age_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_education_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_employment_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_home_adults_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_home_children_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_homeowner_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_income_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_location_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_sex_label (users.tests.test_forms.UserDemographicsFormTest) ... ok
test_email_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_first_name_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_last_name_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_password1_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_password2_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_username_label (users.tests.test_forms.UserRegisterFormTest) ... ok
test_first_name_label (users.tests.test_forms.UserUpdateFormTest) ... ok
test_last_name_label (users.tests.test_forms.UserUpdateFormTest) ... ok
test_age_label (users.tests.test_models.DemographicsTest) ... ok
test_age_name (users.tests.test_models.DemographicsTest) ... ok
test_education_label (users.tests.test_models.DemographicsTest) ... ok
test_education_name (users.tests.test_models.DemographicsTest) ... ok
test_employment_label (users.tests.test_models.DemographicsTest) ... ok
test_employment_name (users.tests.test_models.DemographicsTest) ... ok
test_home_adults_label (users.tests.test_models.DemographicsTest) ... ok
test_home_adults_name (users.tests.test_models.DemographicsTest) ... ok
test_home_children_label (users.tests.test_models.DemographicsTest) ... ok
test_home_children_name (users.tests.test_models.DemographicsTest) ... ok
test_homeowner_label (users.tests.test_models.DemographicsTest) ... ok
test_homeowner_name (users.tests.test_models.DemographicsTest) ... ok
test_income_label (users.tests.test_models.DemographicsTest) ... ok
test_income_name (users.tests.test_models.DemographicsTest) ... ok
test_location_label (users.tests.test_models.DemographicsTest) ... ok
test_location_name (users.tests.test_models.DemographicsTest) ... ok
test_object_format (users.tests.test_models.DemographicsTest) ... ok
test_sex_label (users.tests.test_models.DemographicsTest) ... ok
test_sex_name (users.tests.test_models.DemographicsTest) ... ok
test_updated_label (users.tests.test_models.DemographicsTest) ... ok
test_updated_name (users.tests.test_models.DemographicsTest) ... ok
test_user_label (users.tests.test_models.DemographicsTest) ... ok
test_user_name (users.tests.test_models.DemographicsTest) ... ok
test_banned_label (users.tests.test_models.ProfileTest) ... ok
```

```
test_banned_value (users.tests.test_models.ProfileTest) ... ok
test_bio_label (users.tests.test_models.ProfileTest) ... ok
test_bio_value (users.tests.test_models.ProfileTest) ... ok
test_consent_label (users.tests.test_models.ProfileTest) ... ok
test_deleted_date_label (users.tests.test_models.ProfileTest) ... ok
test_image_label (users.tests.test_models.ProfileTest) ... ok
test_language_label (users.tests.test_models.ProfileTest) ... ok
test_language_value (users.tests.test_models.ProfileTest) ... ok
test_last_visit_label (users.tests.test_models.ProfileTest) ... ok
test_object_format (users.tests.test_models.ProfileTest) ... ok
test_timezone_label (users.tests.test_models.ProfileTest) ... ok
test_user_label (users.tests.test_models.ProfileTest) ... ok
test_user_name (users.tests.test_models.ProfileTest) ... ok
test_user_organisation_justification_label (users.tests.test_models.ProfileTest) ... ok
test_user_organisation_justification_name (users.tests.test_models.ProfileTest) ... ok
test_user_organisation_label (users.tests.test_models.ProfileTest) ... ok
test_user_organisation_value (users.tests.test_models.ProfileTest) ... ok
test_user_position_in_organisation_label (users.tests.test_models.ProfileTest) ... ok
test_user_position_in_organisation_value (users.tests.test_models.ProfileTest) ... ok
test_user_type_label (users.tests.test_models.ProfileTest) ... ok
test_user_type_name (users.tests.test_models.ProfileTest) ... ok
test_edit_user_context (users.tests.test_views.TestViews) ... ok
test_edit_user_get_request (users.tests.test_views.TestViews) ... ok
test_edit_user_if_not_logged_in (users.tests.test_views.TestViews) ... ok
test_edit_user_logged_in (users.tests.test_views.TestViews) ... ok
test_edit_user_post_request_email (users.tests.test_views.TestViews) ... ok
test_edit_user_post_request_password (users.tests.test_views.TestViews) ... ok
test_edit_user_post_request_profile (users.tests.test_views.TestViews) ... ok
test_edit_user_template_used (users.tests.test_views.TestViews) ... ok
test_logged_in_uses_correct_template (users.tests.test_views.TestViews) ... ok
test_login (users.tests.test_views.TestViews) ... ok
test_response_is_json_user_consent (users.tests.test_views.TestViews) ... ok
test_user_consent_context (users.tests.test_views.TestViews) ... ok
test_user_consent_context_if_error (users.tests.test_views.TestViews) ... ok
test_user_consent_url (users.tests.test_views.TestViews) ... ok
test_user_delete_get_request (users.tests.test_views.TestViews) ... ok
test_user_delete_if_not_logged_in (users.tests.test_views.TestViews) ... ok
test_user_demographics_context (users.tests.test_views.TestViews) ... ok
test_user_demographics_get_request (users.tests.test_views.TestViews) ... ok
test_user_demographics_if_not_logged_in (users.tests.test_views.TestViews) ... ok
test_user_demographics_logged_in (users.tests.test_views.TestViews) ... ok
test_user_demographics_post_request (users.tests.test_views.TestViews) ... ok
test_user_demographics_template_used (users.tests.test_views.TestViews) ... ok
test_user_settings_context (users.tests.test_views.TestViews) ... ok
test_user_settings_get_request (users.tests.test_views.TestViews) ... ok
test_user_settings_if_not_logged_in (users.tests.test_views.TestViews) ... ok
test_user_settings_logged_in (users.tests.test_views.TestViews) ... ok
test_user_settings_post_request (users.tests.test_views.TestViews) ... ok
test_user_settings_template_used (users.tests.test_views.TestViews) ... ok

----------------------------------------------------------------------
Ran 306 tests in 4.928s
```

```
OK
Destroying test database for alias 'default' ('test_evident_postgres')...
```

9.2 Appendix 2: User experience evaluation metrics questionnaire (UxEM) for the EVIDENT platform

Topic: Few questions about you	
What your background is about?	[ ] Information and Communications Technology [ ] Bevarioural sciences [ ] Economics
How comfortable are you with using computers for various tasks?	** Likert Scale from 1 to 5* [1] Not comfortable at all [ ] [5] Very comfortable
Topic: Scenarios-based tasks	
Scenario 1: Please, register to the EVIDENT platform as an "organisation".	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed
Scenario 2: While logged in, fill in your demographics.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed
Scenario 3: While logged in, find a text that describes how your data are used in the EVIDENT platform.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed
Scenario 4: While logged in, find the page that describes how you can use the EVIDENT platform to design and implement a lab experiment.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed
Scenario 5: While logged in, create and save a simple questionaire using the corresponding editor.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed
Scenario 6: While logged in, publish your questionnaire by creating a new session.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] What was the steps you followed? Please describe them using bullet points.
Scenario 7: While logged in, copy your session url and by using an incognito browser fill it in.	[Yes/No] Successful [ ] Time needed (in seconds)
Scenario 8: While logged in, download the data collected in xlsx format.	[Yes/No] Successful [ ] Time needed (in seconds) [ ] Clicks needed

Table 6: User experience evaluation metrics questionnaire (UxEM) for the EVIDENT platform

9.3 Appendix 3: Questionnaire for user interface satisfaction (QUIS) for the EVIDENT platform

Topic: Overall reactions to the EVIDENT platform		
** Likert Scale from 1 (terrible) to 10 (wonderful)*		
terrible	1 2 3 4 5 6 7 8 9 10	**wonderful**
** Likert Scale from 1 (difficult) to 10 (easy)*		
difficult	1 2 3 4 5 6 7 8 9 10	**easy**
** Likert Scale from 1 (frustrating) to 10 (satisfying)*		
frustrating	1 2 3 4 5 6 7 8 9 10	**satisfying**
** Likert Scale from 1 (inadequate power) to 10 (adequate power)*		
inadequate power	1 2 3 4 5 6 7 8 9 10	**adequate power**
** Likert Scale from 1 (dull) to 10 (stimulating)*		
dull	1 2 3 4 5 6 7 8 9 10	**stimulating**
** Likert Scale from 1 (rigid) to 10 (flexible)*		
rigid	1 2 3 4 5 6 7 8 9 10	**flexible**

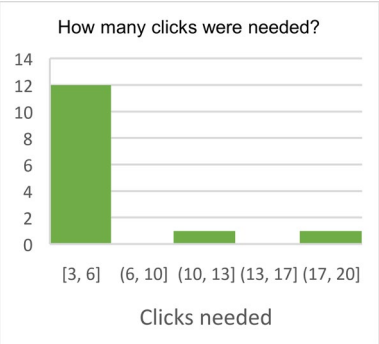
Topic: EVIDENT platform screen factors	
Characters on the computer screen	Likert Scale from 1 (hard to read) to 10 (easy to read)
Highlighting on the screen simplifies task	Likert Scale from 1 (not at all) to 10 (very much)
Organization of information on screen	Likert Scale from 1 (confusing) to 10 (very clear)
Sequence of screens	Likert Scale from 1 (confusing) to 10 (very clear)
Topic: Terminology and EVIDENT platform information	
Use of terms throughout EVIDENT Platform	Likert Scale from 1 (inconsistent) to 10 (consistent)
Computer terminology is related to the task you are doing	Likert Scale from 1 (never) to 10 (always)
Position of messages on screen	Likert Scale from 1 (inconsistent) to 10 (consistent)
Messages on screen which prompt user for input	Likert Scale from 1 (confusing) to 10 (clear)
Computer keeps you informed about what it is doing	Likert Scale from 1 (never) to 10 (always)
Error messages	Likert Scale from 1 (unhelpful) to 10 (helpful)
Topic: EVIDENT platform learning	
Learning to operate the EVIDENT Platform	Likert Scale from 1 (difficult) to 10 (easy)
Exploring new features by trial and error	Likert Scale from 1 (difficult) to 10 (easy)
Remembering names and use of commands	Likert Scale from 1 (difficult) to 10 (easy)
Tasks can be performed in a straight-forward manner	Likert Scale from 1 (never) to 10 (always)

Help messages on the screen	Likert Scale from 1 (unhelpful) to 10 (helpful)
Supplemental reference materials	Likert Scale from 1 (confusing) to 10 (clear)
Topic: EVIDENT platform capabilities	
EVIDENT Platform speed	Likert Scale from 1 (too slow) to 10 (fast enough)
EVIDENT Platform reliability	Likert Scale from 1 (unreliable) to 10 (reliable)
Correcting your mistakes	Likert Scale from 1 (difficult) to 10 (easy)
Experienced and inexperienced users' needs are taken into consideration	Likert Scale from 1 (never) to 10 (always)
Topic: EVIDENT platform usability and user interface (UI)	
Use of colors and sounds	Likert Scale from 1 (poor) to 10 (good)
EVIDENT Platform feedback on user actions	Likert Scale from 1 (poor) to 10 (good)
EVIDENT Platform response to errors	Likert Scale from 1 (awkward) to 10 (gracious)
EVIDENT Platform messages and reports	Likert Scale from 1 (poor) to 10 (good)
EVIDENT Platform clutter and UI "noise"	Likert Scale from 1 (poor) to 10 (good)
Topic: Platform acceptability	
After your experience with the EVIDENT platform, how likely is to use the platform to design and implement an online lab experiment?	Likert Scale from 1 (Not a all) to 10 (I will definitely use the platform)
After your experience with the EVIDENT platform, how likely is to propose the platform to a colleague of yours?	Likert Scale from 1 (Not likely at all) to 10 (I will definitely propose the platform)

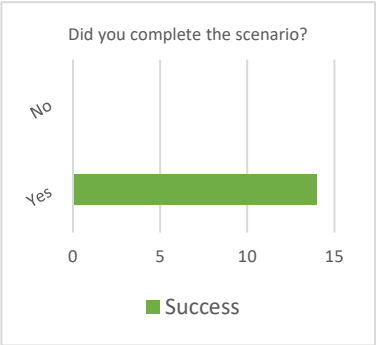
Table 7: Questionnaire for user interface satisfaction (QUIS) for the EVIDENT platform

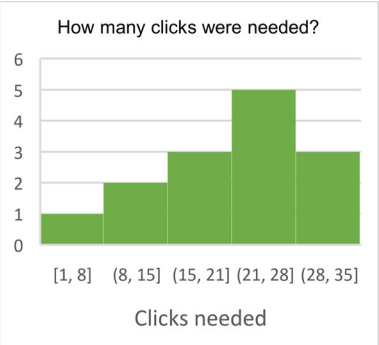
9.4 Appendix 4: User experience evaluation metrics questionnaire individual results

Scenario 1: Please, register to the EVIDENT platform as an "organisation".



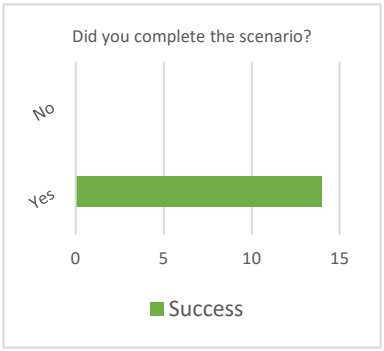
Scenario 2: While logged in, fill in your demographics.

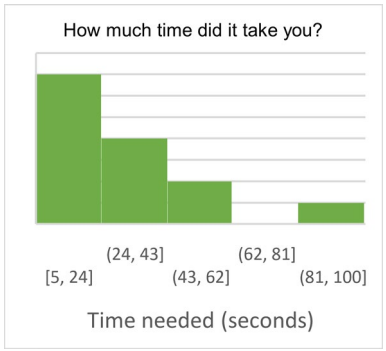


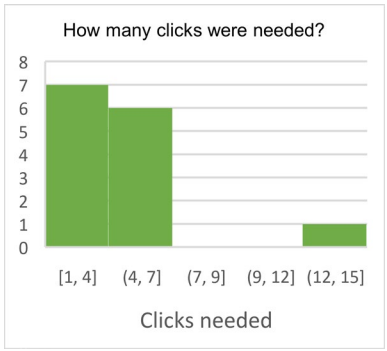


Scenario 3: While logged in, find a text that describes how your data are used in the EVIDENT platform.

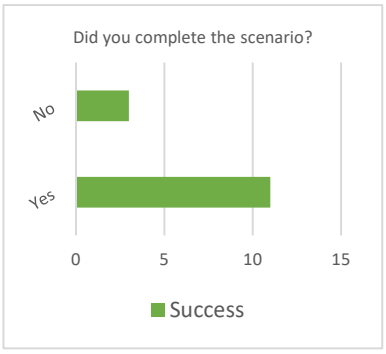
Scenario 4: While logged in, find the page that describes how you can use the EVIDENT platform to design and implement a lab experiment.

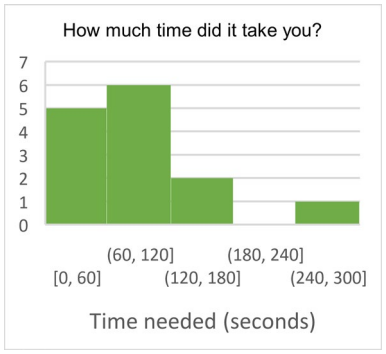


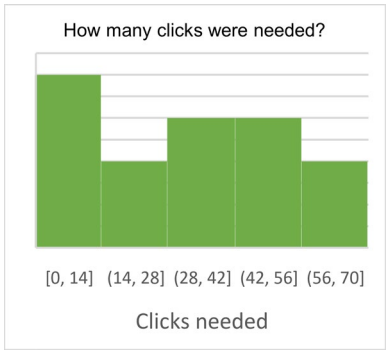




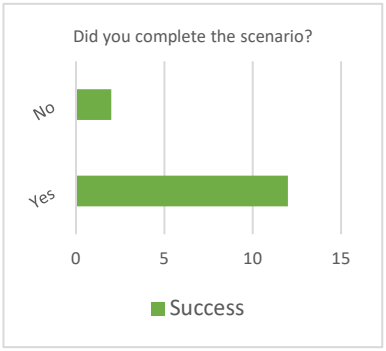
Scenario 5: While logged in, create and save a simple questionnaire using the corresponding editor.







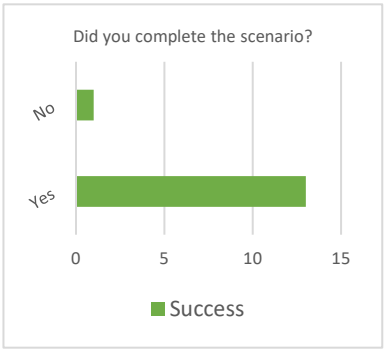
Scenario 6: While logged in, publish your questionnaire by creating a new session.

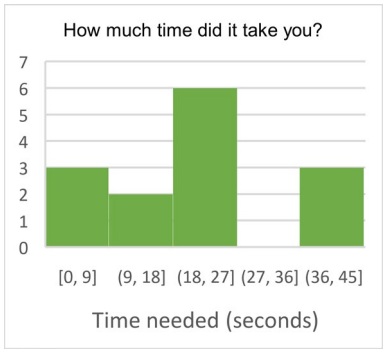


What was the steps you followed? Please describe them using bullet points:

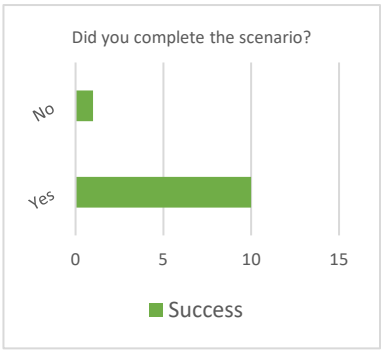
1. First I have created a simple survey, as requested by scenario 5.
2. Then i moused over my profile name and clicked on "my sessions"
3. I chose create new
4. I filled in the title, description and password fields
5. I attached my survey to the current session by selecting "add new application" -> "Questionnaire" and then clicking on my survey's title.
6. I clicked on the "create new session" button

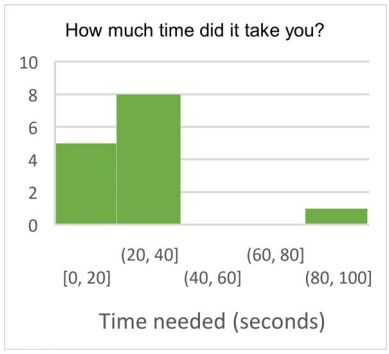
Scenario 7: While logged in, copy your session url and by using an incognito browser fill it in.





Scenario 8: While logged in, download the data collected in xlsx format.





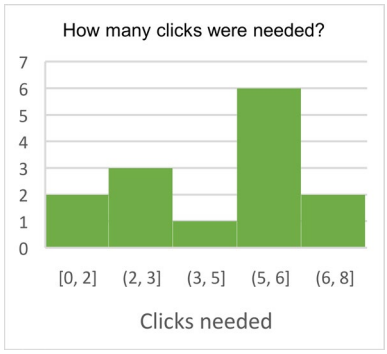
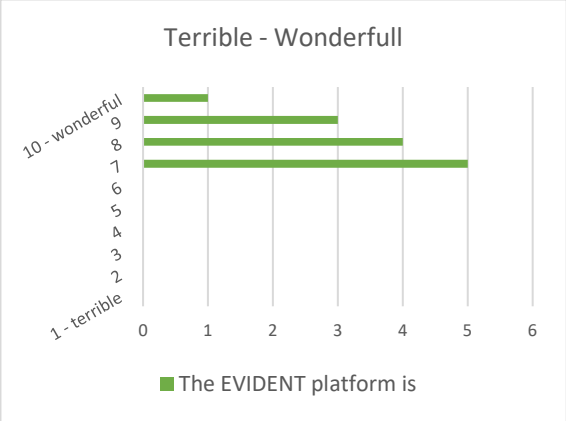
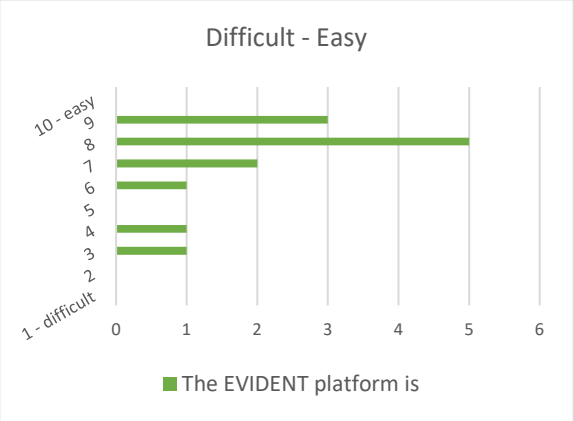


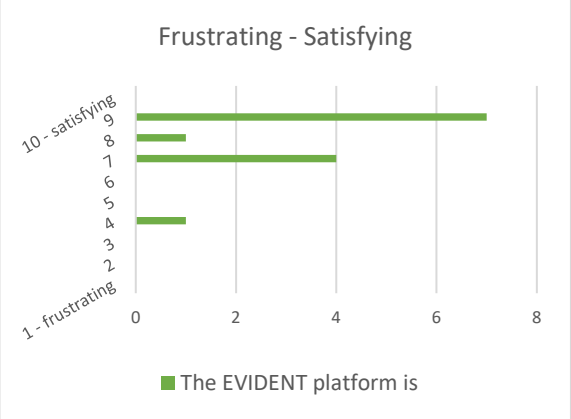
Table 8: User experience evaluation metrics questionnaire individual results

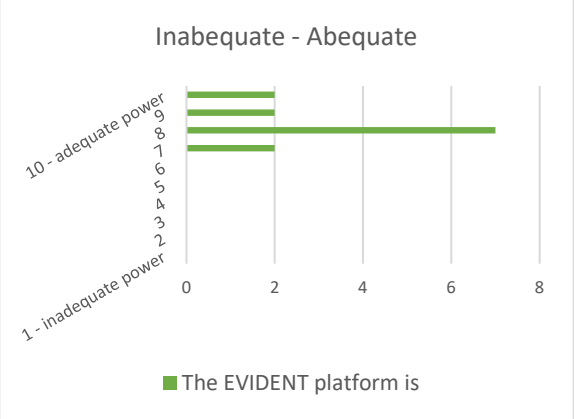
9.5 Appendix 5: User experience evaluation metrics questionnaire individual results

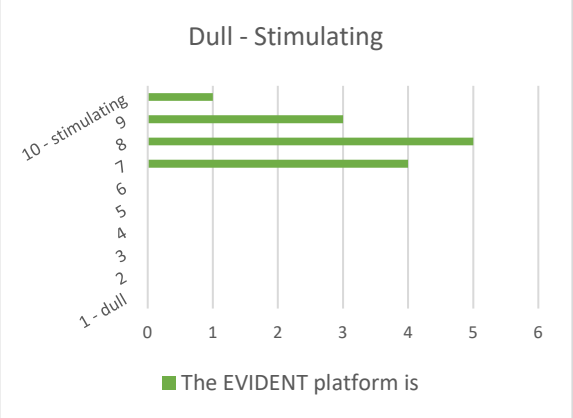
Topic: Overall reactions to the EVIDENT platform

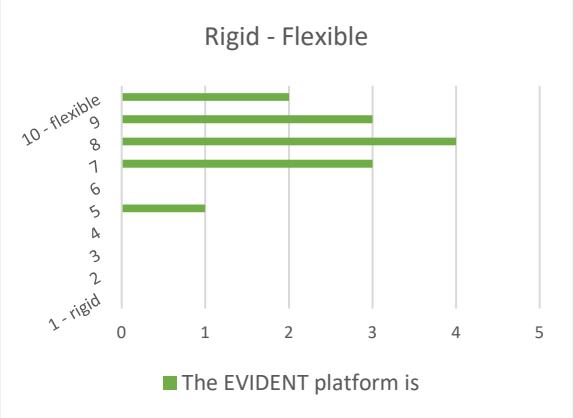




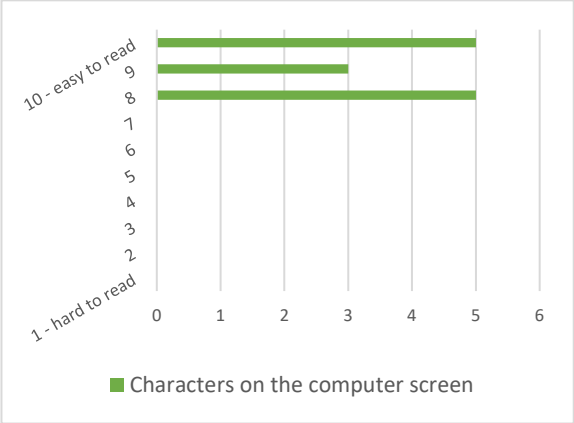


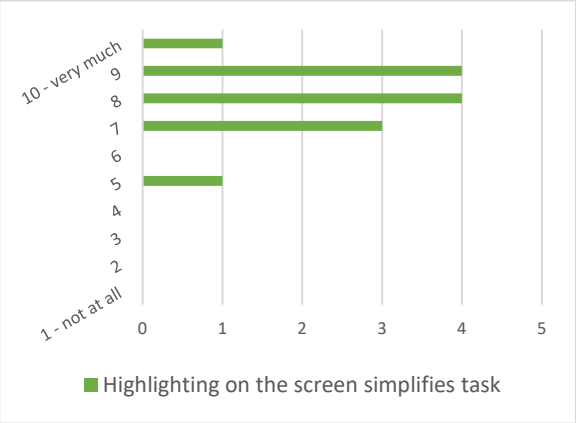


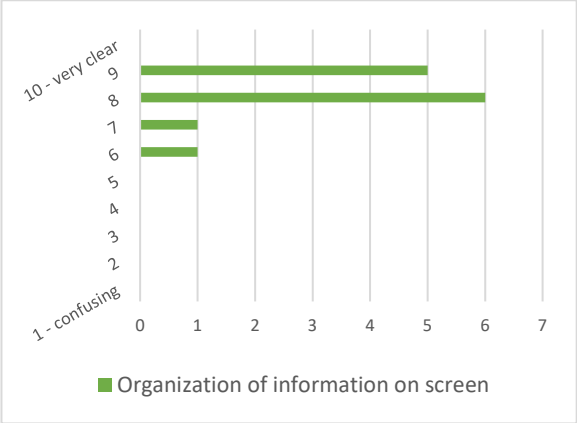


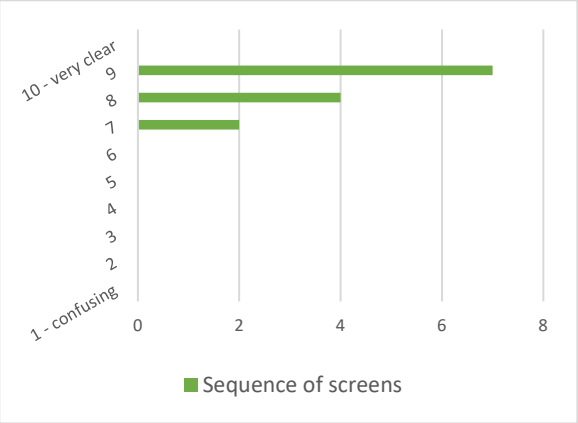


Topic: EVIDENT platform screen factors

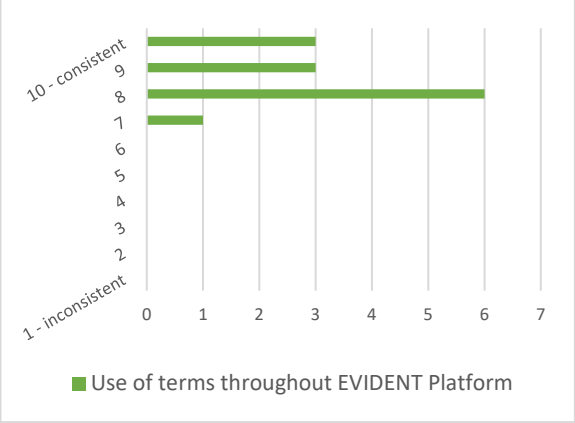


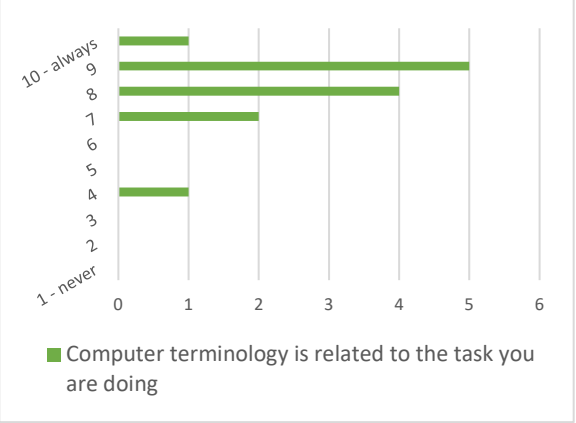


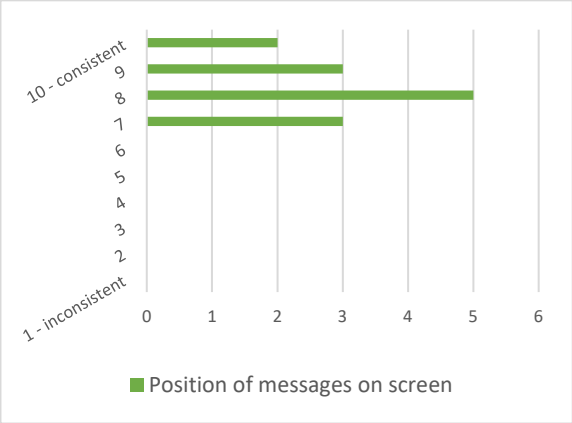


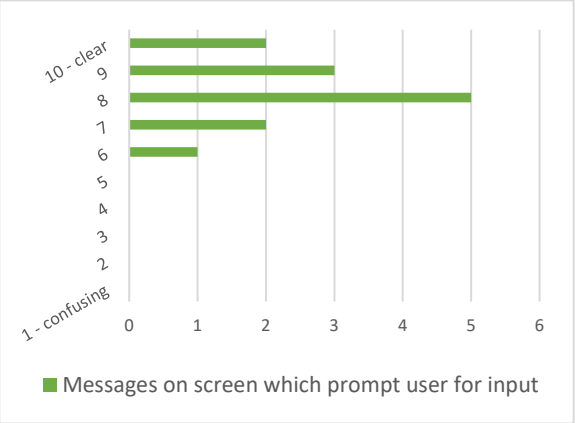


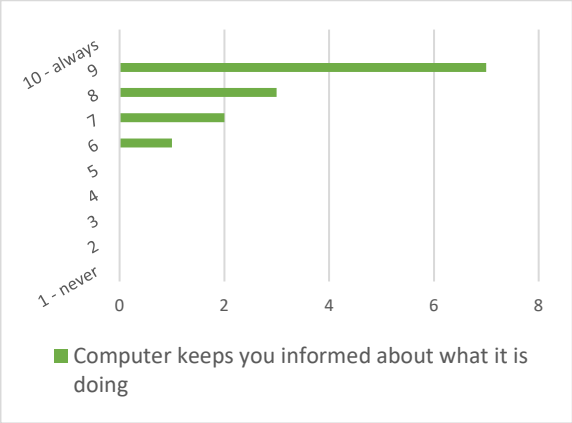
Topic: Terminology and EVIDENT platform information

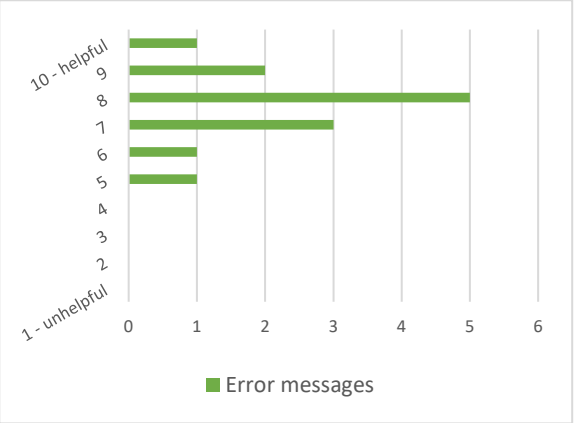




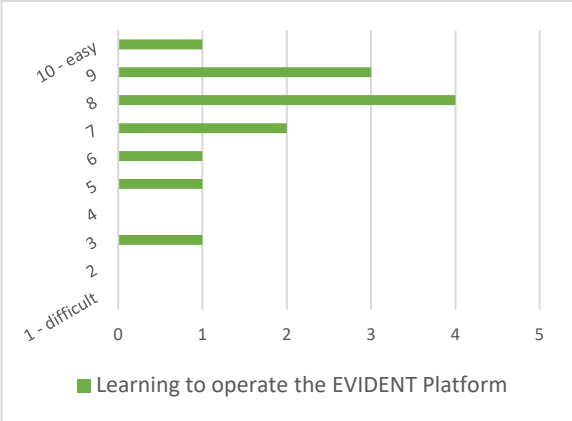


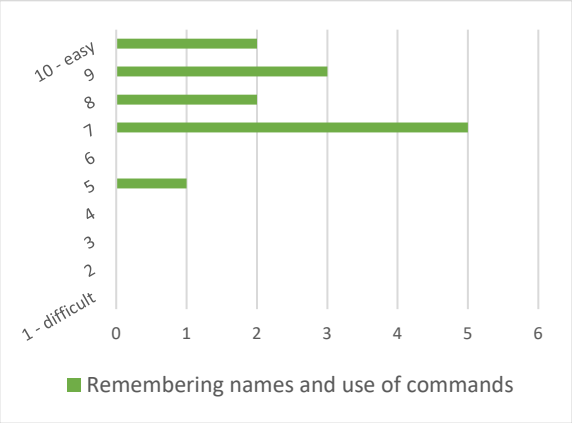


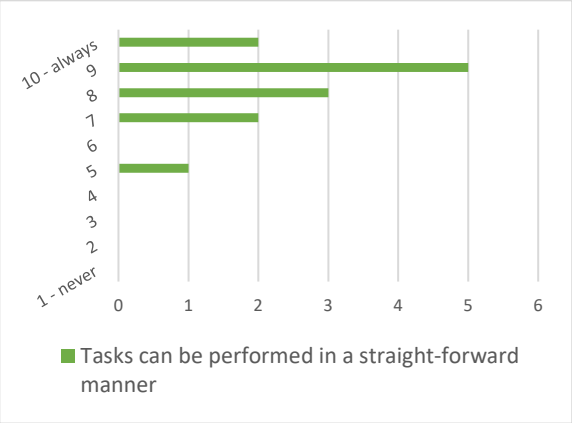


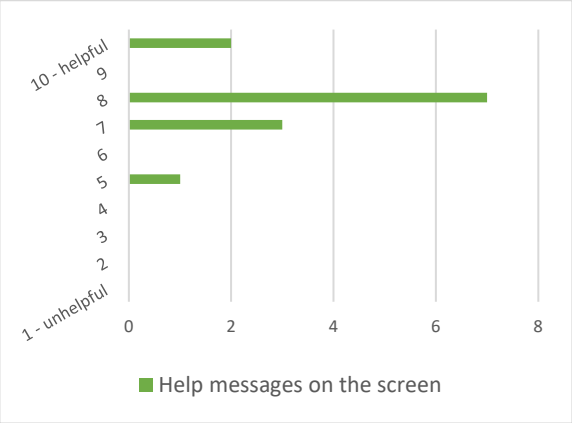


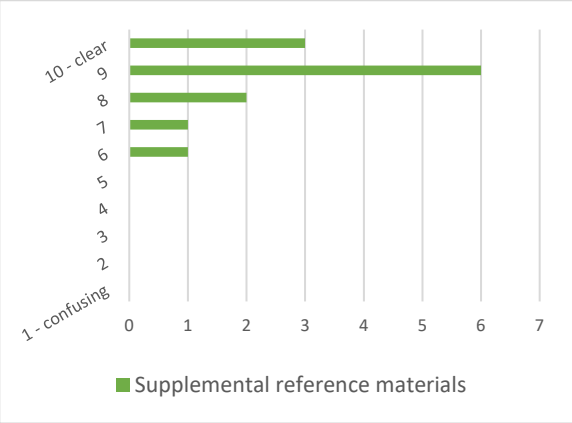
Topic: EVIDENT platform learning



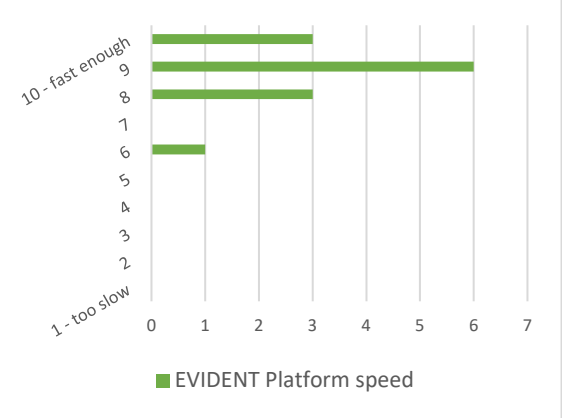


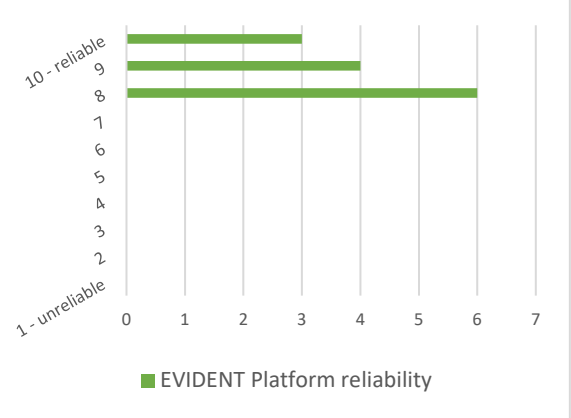


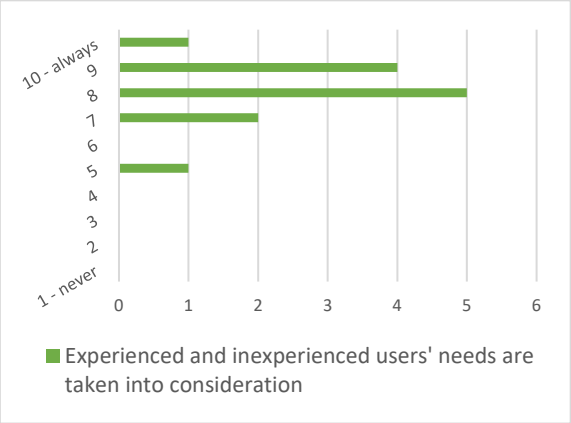




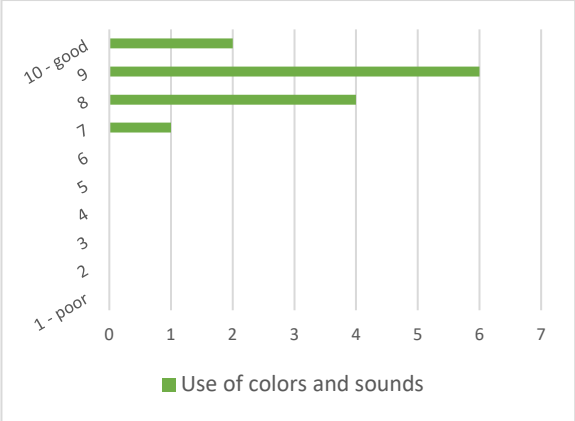
Topic: EVIDENT platform capabilities

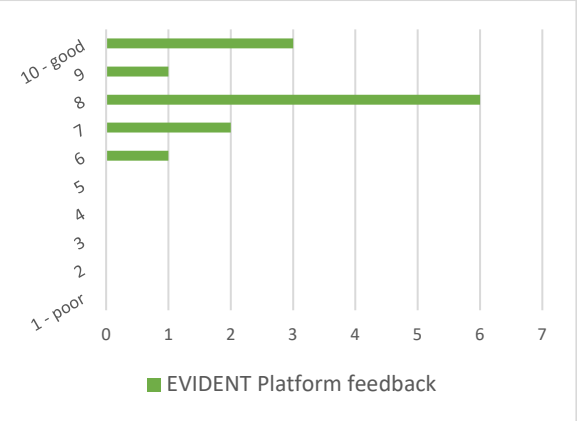


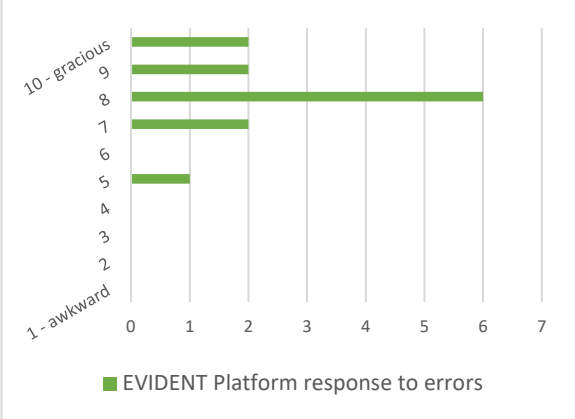


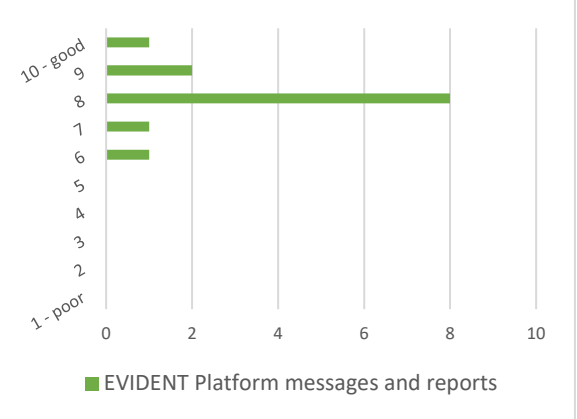


Topic: EVIDENT platform usability and UI









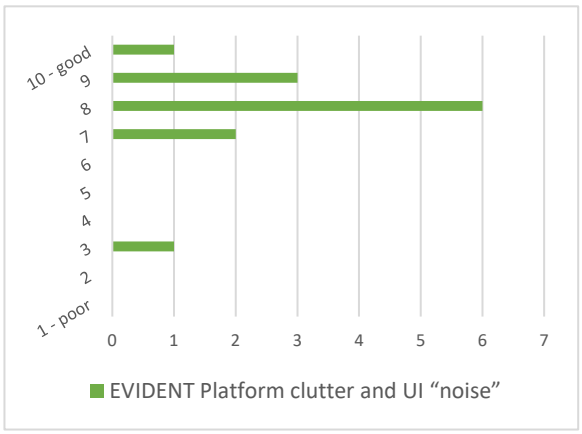


Table 9: User experience evaluation metrics questionnaire individual results