

École polytechnique de Louvain

User Interface Requirements Elicitation by Crowdsourcing

Author: **Andrew ARNITA**
Supervisor: **Jean VANDERDONCKT**
Readers: **Emilio INSFRAN, Mehdi OUSMER**
Academic year 2022–2023
Master [60] in Computer Science

Abstract

As technology advances, effectively understanding user interface requirements remains a significant challenge. This master thesis aims to define, develop, and evaluate *Crowditivity*, a platform that has been designed to use *crowdsourcing* for eliciting user interface requirements by combining the requirements specification template known as *VOLERE*, and the annotations of images related to the design of a user interface that are recognized using the point-cloud recognizer, *\$P+*. *Crowditivity* is a tool used to facilitate the requirements gathering process by using a more user-centered approach, this gives the stakeholders a more practical way of understanding the users' expectations regarding a proposed software.

Acknowledgements

In the beginning, I would like to thank my thesis supervisor Professor J. Vanderdonckt, who gave me the opportunity to work on this thesis and provided me with great guidance throughout its realization. The experience I have gained under their mentorship has been very helpful in completing this thesis.

A big thank you to Prof. Dr. Emilio Insfran and Mehdi Ousmer, who have offered their expertise as readers for this thesis.

A general thank you to all the participants who gave their personal opinions in the studies that were conducted.

Lastly, I would like to thank my family, their belief in me has kept my spirit and motivation high during this process.

Contents

1	Introduction	4
1.1	Context of the Problem	4
1.2	Mission Statement	4
1.2.1	Statement	4
1.2.2	Research Method	5
1.3	Thesis Overview	5
2	Related Work	7
2.1	Software Development Life Cycle (SDLC)	8
2.1.1	Requirements Engineering	9
2.1.2	Requirements Engineering for User Interface	11
2.1.3	Elicitation of User Interface Requirements	12
2.2	The Crowdesign Engine	13
2.2.1	Crowdlicit	13
2.2.2	Crowdsensus	15
2.3	<i>VOLERE</i>	17
2.3.1	<i>VOLERE</i> - SHELL Template	18
2.4	ReqView	19
2.5	Gelicit	21
2.5.1	Meta-model	22
2.5.2	The 6 Steps Process	23
2.5.3	The Software	25
2.5.4	Model-Driven Engineering (MDE)	27
2.5.5	ReDSeeDS	28
2.5.6	Other Tools: SnappView	30
2.6	Limitations of Existing Tools	31
2.7	Conclusion	32
3	Eliciting and Identifying Annotation Types	33
3.1	Definition of Annotations	33
3.2	Elicitation Study of Annotation Types	34
3.2.1	Questions Asked	35
3.2.2	Participants' Information	37
3.2.3	Classification of Data	40
3.2.4	AGATe	43
3.3	Identification Study: Annotation Types	46
3.3.1	Survey Questions	46
3.3.2	Participants' Information	49

3.3.3	Results	52
3.4	Conclusion	54
4	Development of <i>Crowditivity</i>	55
4.1	<i>Crowditivity</i>	55
4.2	Conceptualization	56
4.2.1	The Process	57
4.2.2	Create Study	57
4.2.3	Annotations	58
4.2.4	Review the work	59
4.3	Specification	59
4.3.1	Class Diagram	59
4.3.2	Activity Diagrams	61
4.4	Implementation	66
4.4.1	Technology Stack	66
4.4.2	<i>Crowditivity</i> Walkthrough	69
4.4.3	The $\$P+$ Recognizer	74
4.5	Conclusion	77
5	Putting <i>Crowditivity</i> to the Test: A Case Study	78
5.1	User Experience (UX)	78
5.2	Case Selection	79
5.3	Evaluating <i>Crowditivity</i>	80
5.3.1	Steps guide for creating requirements	81
5.3.2	Steps guide to annotate images	82
5.4	UEQ+	85
5.4.1	Defining <i>UEQ+</i>	85
5.4.2	The Questionnaire	85
5.4.3	Results of the <i>UEQ+</i>	87
5.5	Conclusion	90
6	Conclusion and Future Work	91
6.1	Review	91
6.2	Critical Analysis	92
6.3	Future Work	92
6.3.1	Short-term	92
6.3.2	Mid-term	92
A	General Note	99
B	Results of the Elicitation Study on Annotation Types	100
C	Classification Groups for each Annotation Type	109
D	Annotation Types Recognition on <i>Crowditivity</i>	117
E	Examples of Utilized Forms and Surveys	127
E.0.1	QuestionPro	127
E.0.2	Forms.app	130
E.0.3	Google Forms	135

F	GDPR Consent	140
G	User Annotations from Case Study	147

Chapter 1

Introduction

1.1 Context of the Problem

Designing a good user interface enhances the usability and enables a smooth interaction between the user and the technology that they use. The development of an effective user interfaces highly depends on the quality of the requirements where the end-user should be actively involved [33]. Designers must understand users' needs by gathering accurate requirements in the design process, and an efficient way to do this is by *crowdsourcing*, which has the ability to collect ideas from a group of users having each their own unique perspectives. Using requirements gathered from a crowd of users allows designers to discover new requirements, while also having the ability to gather feedback that would help improve the design of an interface. The main benefit of *crowdsourcing* is, that the issues with the designs are quickly identified, allowing faster adjustments before the final stages of development. Therefore, by using the perspectives of the crowd, we can create interfaces that are more appealing to a wide range of users, ensuring better user interfaces that would improve the overall user experience.

1.2 Mission Statement

1.2.1 Statement

The central goal of this thesis is to define, design, implement, and test a crowd-based software environment for conducting requirements engineering about the graphical user interface of an information system. The requirements should be graphically annotated via appropriate representations that the environment should be able to recognize to properly manage them.

We start our work by giving an overview of the existing research related to the field of user interface requirements. Then, we gather multiple perspectives through an elicitation study and identification survey, which would be used in the development of our platform, *Crowditivity*. After this, we will talk about the process of selecting the right annotation types based on the opinions of a crowd of users to ensure that they are created in the right way. We will then integrate the data collected on these annotations into *Crowditivity*, while having the ability for these types to be recognized using *\$P+*, a point cloud recognizer. As a last step, we will deploy our platform and let a group of users evaluate its performance, usability, and attractiveness and give their feedback using the *UEQ+* assessment tool.

1.2.2 Research Method

Throughout our research, we conducted a series of steps that we used as a roadmap, helping us maintain a good approach towards our mission. By following these steps, we were able to maintain a clear focus on our objectives, identify key areas that require further exploration, and address potential challenges.

1. **Gathering Data:** Before any work, we need to understand the process behind the elicitation studies, what information is needed from the crowd, and what best practices are used to analyze the results. To do that, we have analyzed studies related to this process, such as: "*Gesture Elicitation Study*" and "*Crowdsourcing Similarity Judgments*".
2. **Data Analysis:** Then, we examine user responses to find similarities that give us information about their preferences. By analyzing the data, we aim to have valuable information that will help us conclude that the annotations used meet the users needs.
3. **Development:** Afterwards, we have to develop a platform that would help in the elicitation of user interface requirements as well as giving the ability for the user to annotate on images that are included in a design process and having the ability to recognize these annotations.
4. **Case Study:** Finally, we will conduct a case study that puts our platform to the test in a real world scenario. By letting participants interact with our platform, providing feedback on its usability, the case study will serve as an assessment of various aspects of the user experience spectrum, such as the user interface, ease of use, and the ability to elicit requirements. Users that test our platform will participate in our *User Experience Questionnaire* to give their feedback on their experience with *Crowditivity*.

1.3 Thesis Overview

To meet the objectives of this master thesis, we will define each step of our process in its own chapter. The following chapters will elaborate on our work, from initial discoveries to final conclusions.

Chapter 2, referred to as “**Related Work**”, is designed to explain the work that was performed in terms of user interface requirements elicitation. It is an exploration of different tools and methodologies that each contributed to the process of gathering requirements.

Chapter 3, referred to as “**Eliciting and Identifying Annotation Types**”, is focused on defining the annotation types that will be applied on a user interface. We collected different interpretations of the annotation types and surveyed users on their preferred type through a series of tasks. After that, we will discuss the process of how we classified the data and the use of *AGATe*, a tool that calculates the consensus among several people working on the same dataset.

Chapter 4, referred to as “**Development of *Crowditivity***”, will explain the development process of *Crowditivity*, our platform that we built to gather user interface requirements. In this chapter, we are going to explore the steps that guided its development, discussing the important requirements and the tech stack that we used. To conclude this chapter, we will talk about the implementation of $SP+$, and how this algorithm was used to recognize user-generated annotations on images.

Chapter 5, referred to as “**Putting *Crowditivity* to the Test: A Case Study**”, will showcase our platform’s performance in a real-world scenario, identifying its capabilities and any potential improvements. We will discuss how we conducted the case study, the data we gathered, and the results of our *UEQ+*.

Finally, in chapter 6, referred to as “**Conclusion and Future Work**”, we will summarize the work that was done in this thesis and highlight the important things that we gained from our findings. Furthermore, we will discuss how these results could impact future works in figuring out what are extra features that can be added to *Crowditivity* to better enhance its capabilities.

Chapter 2

Related Work

This chapter is focused on the work that has been done on user interface requirements elicitation that assisted us in writing this master thesis. First, in Section 2.1, we will introduce the Software Development Life Cycle (SDLC), discuss how to develop meaningful requirements, and discuss the process of requirements engineering with a specific emphasis on requirement engineering for the user interface. In Section 2.2, we will introduce the Crowdesign engine, what Crowdsensus and Crowdlicit are used for, and how they work hand in hand to help researchers gather requirements efficiently. Section 2.3 focuses on what *VOLERE* means and how its *SHELL* template is being used to solve the communication problem between stakeholders. Section 2.4, is going to introduce *ReqView*, a software that helps with managing *VOLERE* requirements. In Section 2.5, we will see how Gelicit, a cloud platform, was created to elicit gesture requirements. Afterwards, in section 2.6, will be focused on ReDSeeDS, a software solution designed to facilitate the requirements engineering process. Finally, in Section 2.7, the last section of this chapter, we will identify the limitations of the analyzed tools with the goal of addressing and improving upon them in our own platform.

2.1 Software Development Life Cycle (SDLC)

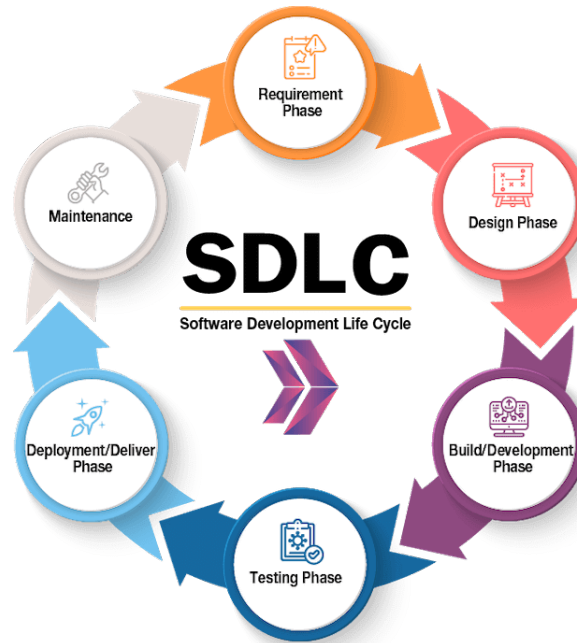


Figure 2.1: Software Development Life Cycle

Software Development Life Cycle (SDLC) methodologies are mechanisms to assure that software meet established requirements [26]. It also suggests improvements for current agile development so that this lightweight SDLC could be adopted more in practice for organizational project management [28]. Figure 2.1 shows how this methodology is divided into phases that a software being developed passes by. The phases are as follows:

1. **Requirement Analysis:** the main focus of this step is to identify what are the client's requirements and documenting it in a clear way. This phase is considered to be the most important in SDLC. It provides an overview about the behavior of the software and what it should look like. At the end of this step, a Software Requirement Specification (SRS) is created as a description of the software's behavior.
2. **Designing:** in this step, we take everything that has been discussed in the first step, analyze it, and start the problem solving journey between developers and designers on how these requirements must be illustrated in a clear manner. The results are described in a document known as the Software Design Document (SDD) that includes all the necessary details and solutions for a specific problem.
3. **Coding:** for this step, the SDD created in the previous phase is transformed into code. Using a programming language defined in the design phase, developers are now focused on developing software that meets the client's requirements. At the end of this phase, the client is presented with a prototype of what the software looks like.
4. **Testing:** this step is crucial after coding step, each section of the software that has been planned and developed must be tested. Testing ensures that the software is free of bugs, works without errors, and behaves as intended. Testing can have

a positive outcome if done properly, such as low maintenance in the future, and produces high-quality software.

5. **Deployment:** also called the production phase. This step is similar to the delivery service, where the software is ready to be shipped to its clients. Developers and testers present their work to the client and explain all the details and important information on how the software works and how each requirement has been implemented.
6. **Maintenance:** this is the final stage of SDLC. At this level, all requirements are met, the software was deployed successfully, and the client is using the software. This step includes patch updates, some bug fixes discovered in the production phase, and if necessary changes that the client might request. Maintenance is an endless loop where you ensure that software still behaves the same way it was delivered.

SDLC has various software development approaches which are defined by models. Some example are: Waterfall model, V-model, Spiral model, Incremental model, etc. Each of these models uses a specific life cycle to ensure that the software development process is successful. The next topic will explain what is requirements engineering and its phases, as well as how we can use such process when gathering requirements for user interfaces.

2.1.1 Requirements Engineering

Requirements Engineering (RE) is, as its name suggests, the engineering discipline of establishing user requirements and specifying software softwares [43]. Requirements engineering is the area of software engineering that deals with the process of developing and verifying software requirements [17]. The software industry has many failures caused by the complexity of software requirements. In the requirements engineering process, we need to identify three main questions: What are the requirements? Who and what are the roles of stakeholders in these requirements? What is the definition of requirements engineering? First, we need to understand what a requirement is in the first place because errors in the requirements gathering and software specification phases lead to project failure and costly mistakes during software development [8]. A requirement is a condition that is needed by a user to solve a specific problem, and to work on this requirement, different stakeholders are involved with their role to contribute to the development:

- **Direct Stakeholders:** are people or can sometimes be organizations, that have an impact on the project. Different types of stakeholders are present throughout the process of achieving a successful project, they are called direct stakeholders because they are directly impacted by the project being implemented, for example, the users of a software.
- **Indirect Stakeholders:** are people who must know the changes that are being made in the project but are not directly impacted; for example, they can be government policies around the rights and regulations of that project.

Now that we covered the important parts of a requirement process, we will discuss furthermore what requirements engineering is meant by. In software engineering, we start by eliciting the requirements from the stakeholders, and then start the software design process. Next, we start the implementation using code, where we will test each part of it before deploying it. The last process in software development is maintenance, where we

would have to maintain our software for a long time eventually. The maintenance process takes much more effort than all the previous steps, and the main reason is because people working on this software did not do a proper job in requirements engineering. Requirements in general, are what would people eventually want in a software, characteristics required by the stakeholders to solve a problem or reach a certain goal. Some of the requirements must deal with constraints, where some may come from software software that a software would eventually need to implement, or technical constraints, for example having to deal with problems later when deploying the software to the cloud. For the engineering part, we generally do things according to a systematic approach. People try to follow some rules and guidelines and most of the time it is a good way to use the best practices that have been used over the years to solve problems. We must differentiate within requirements, typically the functional and non-functional requirements. *Functional* requirements are all the essential requirements that have been discussed with stakeholders and are an essential part of the software being developed to achieve the desired result. On the other hand, *nonfunctional* requirements are not directly related to features in the software, it focuses on things like performance, security, and the robustness of that software.

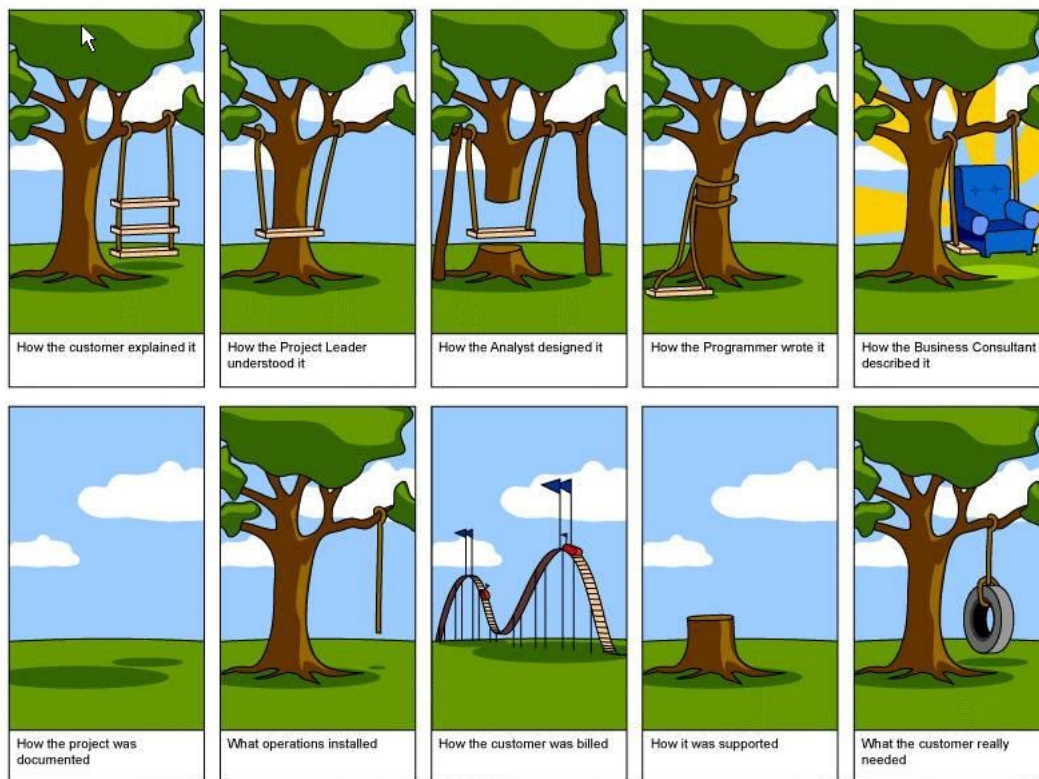


Figure 2.2: Stakeholders' perspective on Requirements

Requirements engineering is done by a requirement engineer that has a core job to translate all the requirements in a way that different stakeholders from the business side of the software and the technical side can understand what is being requested from them based on their roles. To go into more depth about what could be a definition for requirements engineering, we need to understand the main purpose of it. Requirements engineering is defined as a disciplined approach to specifying and managing requirements with specific goals. The process starts by performing a feasibility study to understand the relevant requirements; then we need to reach consensus among the stakeholders on these

requirements, and we finish the process by documenting them according to predetermined standards, to manage them systematically. The core feature of requirements engineering is to fully understand and precisely document the requirements given by stakeholders; this will prevent delivering software that does not meet the required specifications, which will then lead to problems in the development process. We need to perform a series of steps to gain a good understanding of requirements such as elicitation studies, documentation, validation, and management.

- **Feasibility Study:** The main goal of this study is to determine what are the reasons behind the development of this software, how feasible it is, and how adaptive can it be to changes in the future. Different types of feasibility are present when performing this study. The technical part where we analyze the technologies needed to create these requirements on time. Then, we have the economic part, where stakeholders decide how can this software generate financial profits. We conclude this study by doing the operational part, where it focuses on knowing how to perform a series of steps to solve a business problem.
- **Stakeholders' Engagement:** to elicit and better understand their needs and requirements. Elicitation procedures such as workshops and requirement interviews are useful to better understand the stakeholders' needs.
- **Documentation:** After eliciting all the requirements for developing the software, a good documentation of what has been obtained is crucial. Documentation is about writing the requirements in a format that would be clear, complete, and understandable by all the different stakeholders from the business team and the technical team that are working to build this software.
- **Validation:** to be sure that what has been elicited and documented is what is really needed for the development of this software. In technical terms, validation is ensuring that what has been written in a formatted document, is valid and agreed by all the stakeholders, and making sure that what is needed is well understood and clear. By doing the validation process, we are making sure that when we start the development process, what has been documented is agreed on. The validation process prevents problems in the future where it is hard to go back to square one.
- **Changing Requirements:** this step is done to manage any requirement changes that might arise in the future. This is the job of the requirement engineer where it is vital to efficiently keep track of all the changes that might occur in any requirement. When we keep track of requirements, we need to make sure that the constraints between them are well managed and visible to avoid the complexity of having extra constraints when changes are made.

2.1.2 Requirements Engineering for User Interface

A well-crafted user interface enhances the way users interact with a software. This is when the requirements engineering for the user interface comes into action, and this process is needed when software development. Crafting requirements for user interfaces involves a multi-step process of identifying and understanding both functional and non-functional requirements. This process requires an in-depth comprehension of the user's

contextual and environmental factors, which are then used to translate user needs into a comprehensive set of requirements. This typically involves conducting market research, gathering insights from stakeholders and users through interviews, and refining requirements using methodologies such as prototyping and mock-ups. In practice, it is imperative to recognize that requirements engineering for user interfaces is a process that demands attention to detail, clear communication, and active collaboration with the end-user. Although the identification and specification of these requirements may be confusing, the use of research techniques such as market analysis and user interviews can help reduce this and provide insight on how to proceed. Additionally, using techniques such as prototyping and mock-ups can add valuable context to the development process, enabling developers to more effectively tailor their solutions to the user's needs. Ultimately, creating successful user interfaces requires an interdisciplinary approach that draws on insights from design, research, and development. By incorporating a wide range of perspectives and techniques, it is possible to create user interfaces that are both functional and tailored to the specific needs of end-users. By defining requirements, analyzing them and specifying them in a clear and concise way, software developers can create user interfaces that are usable, efficient, and pleasant to use. With a well-defined process and the right techniques, requirements engineering for user interfaces can lead to the creation of software that is effective and more user-friendly.

2.1.3 Elicitation of User Interface Requirements

The lack of user participation and the communication gap between developers and end-users are the main causes of poor requirements and the elicitation of user interface requirements. Likewise, a bad user interface reduces the quality of the software [9]. The focus is on gathering, understanding, and documenting the specific needs and preferences of users regarding the design, appearance, and functionality of a software application or software's user interface. This phase involves a multitude of techniques, including user interviews, surveys, usability testing, and observation of user behaviors, to identify user needs, preferences, and limitations while using the reproduced interface. The information gathered is then transformed into design specifications that serve as a blueprint for the creation of an easily navigable interface that meets the users' expectations. The importance of eliciting user interface requirements lies in its potential to optimize the usability and functionality of the application, thereby ensuring user satisfaction and enhancing the overall user experience. When done correctly, this process can result in higher levels of user engagement, increased productivity, and improved business outcomes. The elicitation process can be beneficial for both designers and developers; By gathering and using user feedback during the user interface design process, designers can create interfaces that are user-friendly and tailored to the needs of users. *Effective elicitation* of user interface requirements is crucial in developing software applications that meet the ever-evolving needs of users. Additionally, including user feedback in the development process enables developers to identify design flaws and areas of improvement, thus reducing development costs and saving time and revisions later in the process. Gathering user requirements can be problematic as users may experience difficulties in expressing their requirements which can cause confusion and gaps in the requirements gathering process, leading to a user interface that fails to meet its users' needs.

2.2 The Crowdesign Engine

This thesis will talk about eliciting requirements using a crowd based software. This kind of elicitation study will change the way we normally interpret requirements, i.e., manual analysis. In this part, we explain what the purpose of the crowdesing engine is, how researchers and practitioners can do an elicitation study in Crowlicit and analyze their results using machine learning algorithms in crowdsensus. Figure 2.3 shows the landing page of the Crowdesign engine platform.

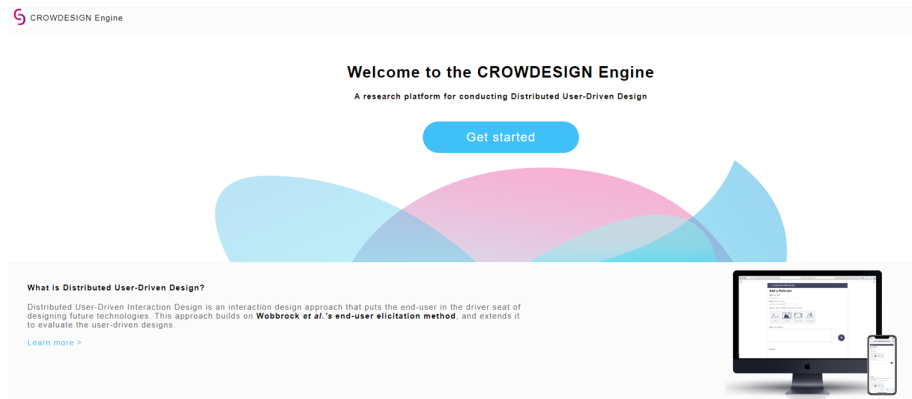


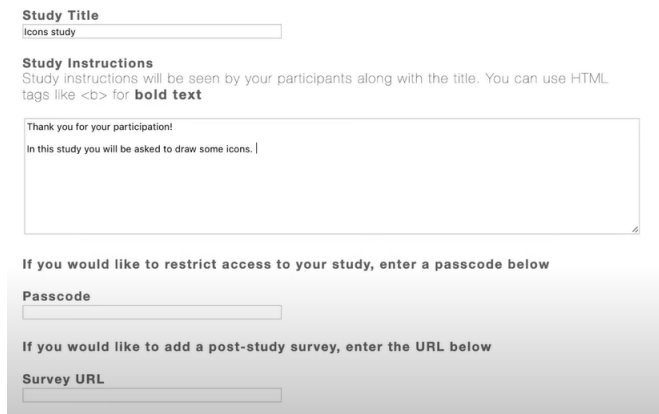
Figure 2.3: Crowdesign Engine

2.2.1 Crowdlcit

We can define crowdlcit as being a software for conducting distributed end-user elicitation and identification studies [4]. The idea behind Crowdlcit is, to demonstrate that elicitation studies can be performed online using a crowd of end-users and still produce quality results in little time. Crowdlcit can be used by anyone and is a unique tool that researchers use to facilitate the gathering process of end-user propositions, such as voice commands, icons, hand gestures, and button labels. Now, we need to understand the process of gathering such propositions from the end-user using the Crowdlcit tool. After understanding what is the purpose of using Crowdlcit, we will walk through the steps required for creating an end-user elicitation study:

- **Create a study:** Figure 2.4 shows the first step in creating an elicitation study. Choosing a relevant name and writing a description explaining the purpose of that study, helps the end-user understand the purpose behind it.
- **Referent choice:** Figure 2.5 shows different types of referents to choose from. A referent is defined as an action to be executed by an end user in a certain context of use [51, 53], that is, opening / closing a file.
- **Symbol type:** Figure 2.6 shows what kind of interaction the user needs based on a specific reference. A symbol is an action that invokes the referents. A symbol can be a voice command, i.e: "*please open/close this file*".
- **User intended questions:** Figure 2.7 shows an example of questions that can be asked to the end-user to rate their proposition. Such questions are optional.

- **Summary:** Figure 2.8 shows a summary of the study that was elicited. Where researchers can start/stop a study, edit referents needed to be collected, and the overall results gathered from the end-user.



Study Title
Icons study

Study Instructions
Study instructions will be seen by your participants along with the title. You can use HTML tags like for **bold text**

Thank you for your participation!
In this study you will be asked to draw some icons. |

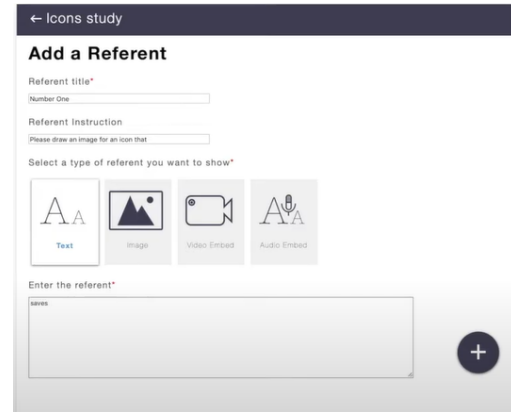
If you would like to restrict access to your study, enter a passcode below

Passcode

If you would like to add a post-study survey, enter the URL below

Survey URL

Figure 2.4: Study Information



← Icons study

Add a Referent

Referent title*
Number One

Referent instruction
Please draw an image for an icon that

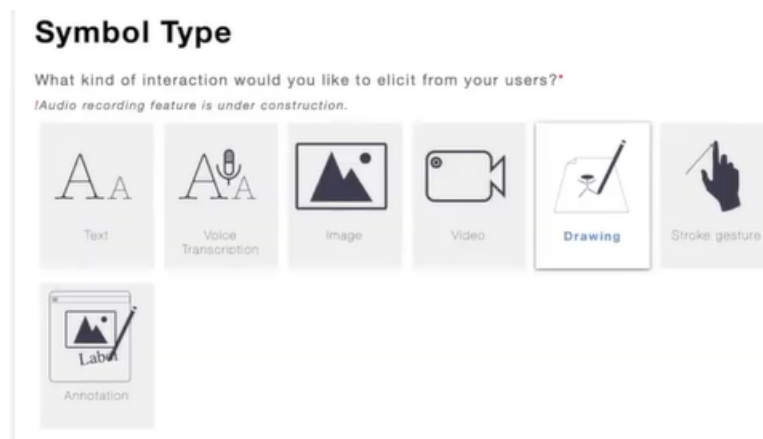
Select a type of referent you want to show*

Text Image Video Embed Audio Embed

Enter the referent*

Save

Figure 2.5: Referent Choice



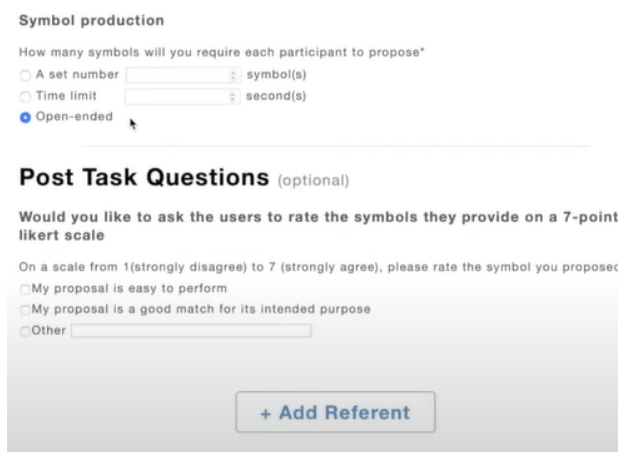
Symbol Type

What kind of interaction would you like to elicit from your users?*

Audio recording feature is under construction.

Text Voice Transcription Image Video Drawing Stroke gesture Annotation

Figure 2.6: Symbol Type



Symbol production

How many symbols will you require each participant to propose*

☐ A set number 5 symbol(s)
☐ Time limit 5 second(s)
☒ Open-ended

Post Task Questions (optional)

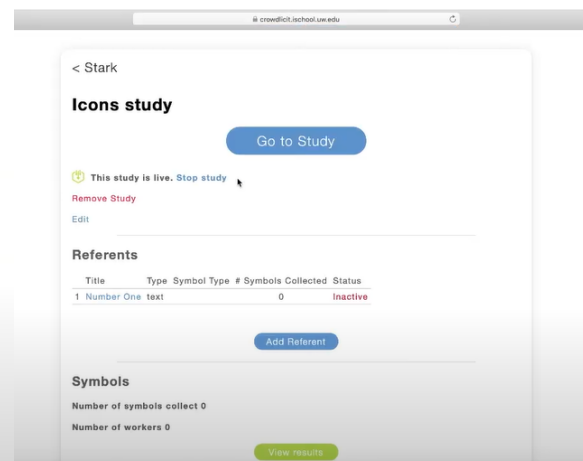
Would you like to ask the users to rate the symbols they provide on a 7-point likert scale

On a scale from 1 (strongly disagree) to 7 (strongly agree), please rate the symbol you proposed

☐ My proposal is easy to perform
☐ My proposal is a good match for its intended purpose
☐ Other

+ Add Referent

Figure 2.7: User Feedback



< Stark

Icons study

Go to Study

This study is live. Stop study

Remove Study

Edit

Referents

Title	Type	Symbol Type	# Symbols Collected	Status
1 Number One	text		0	Inactive

Add Referent

Symbols

Number of symbols collect 0
Number of workers 0

View results

Figure 2.8: Study Summary

In this thesis, our work is focused on building our own platform, where we can elicit specific types of requirements from end-users. As mentioned earlier, the Crowdesign engine has different parts that work together with each other. So, we will understand how the data collected from the end-users is analyzed using a tool called **crowdsensus**.

2.2.2 Crowdsensus

Crowdsensus, part of the Crowdesign engine, is a crowd-based tool. It works hand in hand with the crowd-licit tool described above. Crowdsensus allows researchers to crowdsource crucial similarity judgments central to agreement analysis [3]. This tool has two main parts: the **crowd** and the **algorithms**. We are going to outline what each part does separately and how they help researchers analyze their elicitation studies:

- **Crowd:** Is mainly built of non-experts in the field. Since the end-product is being used by the end-user, researchers focus on these type of users when using Crowdsensus for their studies. Researchers are able to gather as much information as possible and about four times faster than expert researchers [3].
- **Algorithms:** Crowdsensus uses automatic clustering algorithms to analyze data collected from end users. Five optimization algorithms has been developed: steepest-ascent hill climbing, shotgun hill climbing, simulated annealing, a genetic algorithm, and correlation clustering. These algorithms were used to find subgraphs. Figure 2.9 shows all the symbols connected together to create groups of similar symbols, then researchers choose the symbol with the highest consensus among the others.

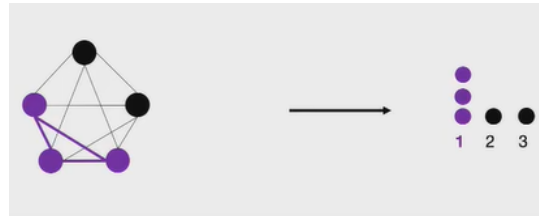


Figure 2.9: Similar Symbols

Crowdsensus is a complete software. It covers the full life cycle from the beginning to the end of an elicitation study. Performing metrics on which symbols are best to use based on a percentage margin of the calculation of the agreement rate, which is done using the formula shown in Figure 2.10 [3]. Without crowdsensus, researchers would have to perform calculations for each referent **manually**.

$$A = \frac{1}{|R|} \sum_{r \in R} \sum_{P_i \subseteq P_r} \left(\frac{|P_i|}{|P_r|} \right)^2$$

Figure 2.10: Agreement Rate

This tool has an original way to interpret symbols. Symbols can be collected by participants, for instance, grouping all similar symbols can be achieved automatically, whereas

in the real world, this would be done manually. Thousands of referents can be present when performing an elicitation study. Therefore, having crowdsensus take care of the calculations automatically helps researchers to be more efficient in their work. Figure 2.11 shows a graph of a real-world example of all possible symbols joined by similarities.

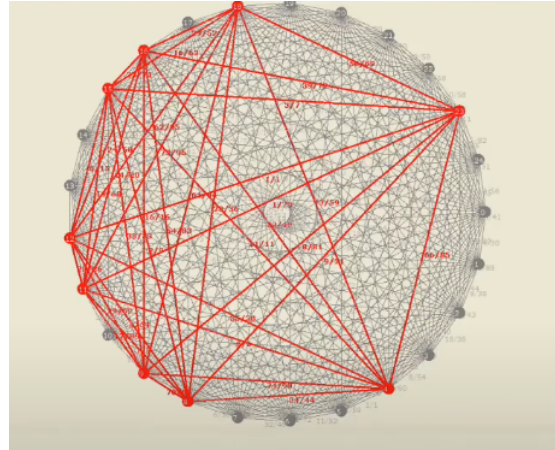


Figure 2.11: Real World Example

Crowdsensus uses *Amazon Mechanical Turk* (AMT). Having a large-scale database such as Amazon's can make a difference and would help reach people faster and more efficiently than sending the elicitation study manually to each end-user. AMT can target a large number of users from different backgrounds and with low cost. When conducting an elicitation study, experts have their own way of thinking. Figure 2.12 shows that non-experts users have similar results to the experts' when eliciting symbols.

Referent <i>How would you phrase a voice command that would...</i>	Experts' symbol	Crowd's symbol
1. open the browser?	"Open browser"	"Open browser"
2. perform a search for cat videos?	"Search for cat videos"	"Search for cat videos"
3. click the first link on the page?	"Click the first link"	"Click the first link"
4. switch between pages in a backward direction?	"Go back 'one / to the last / to the previous' page"	"Go to the previous page"
5. switch between pages in a forward direction?	"Go to the next page"	"Go to the next page"
6. switch between multiple open tabs?	N/A	"Switch tabs"
7. select a region on the page?	N/A	N/A
8. request the same page you are browsing to be loaded again?	"Reload the page"	"Reload the page"
9. bookmark a page?	"Bookmark this page"	"Bookmark this page"
10. close the browser?	"Close browser"	"Close browser"

Figure 2.12: Experts vs Non-Experts

Limitations

Crowdsensus comes with its own set of drawbacks. Earlier, we discussed the useful aspects of why Crowdsensus is unique. The original idea of using Amazon Mechanical Turk (AMT) is beneficial to reach a high number of participants for an elicitation study. When using one of Amazon’s services, at a certain point, the benefits come with a price. Researchers on a budget would have to think twice before using the Crowdsensus tool.

In an elicitation study, the authenticity of the collected data is important. When researchers outsource their studies, they need to make sure that their end-users participating in their study are knowledgeable of what is being requested from them based on their backgrounds. Crowdsensus uses AMT, so end users come from different backgrounds. Researchers may doubt the expertise that this Amazon service provides them.

Crowdlcit includes efficient techniques to validate the results, because having many participants in an elicitation study, validating their votes is a crucial part. Techniques can be used to monitor votes. First, if a user spends more than a specific time on the task, it is considered valid. Then, if the participant did not select the original prompt from the proposed lists of symbols, the tool would detect that this vote is invalid, and it would be removed from the study.

To reach consensus between symbols, they analyze the results using an agreement matrix. Figure 2.13 shows an example of 20 participants who have voted in an elicitation study, where 15 voters agreed that symbols 2 and 4 are similar to each other.

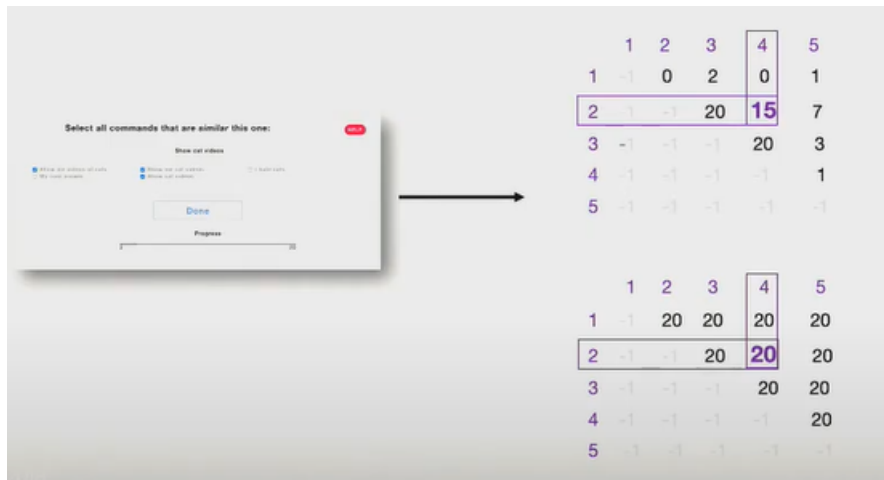


Figure 2.13: Voting Matrix

When building software, multiple stakeholders from different backgrounds have their own role. Having to deal with all the requirements given by many people can make the communication between them difficult, so having a technical language such as *VOLERE*, helps the process of building that software.

2.3 VOLERE

VOLERE is an Italian verb that means: *to wish or to want*. The *VOLERE* language was created by Suzanne and James Robertson. It was created to answer a question asked by

business analysts: "Why can't we build softwares that help us to do our work?" [32]. Most of the time, stakeholders' requirements are unclear and ambiguous. A common problem between different stakeholders must be solved. When a software is built, good code is produced from a developer's perspective, but it did not solve the real business problem. This is where the use of *VOLERE* Requirements Techniques was introduced to fix the problem-solution gap. It has helped the communication between stakeholders from the business side and the developer side to be easier.

VOLERE is a *Socio-Technical* approach:

- **Sociological:** it helps stakeholders coming from different backgrounds, by doing workshops, brainstorming, and apprenticing to try and discover what the requirements are.
- **Technical:** this is a form that ensures that the requirements are understood by everyone involved in the creation of the software. It helps developers and testers understand business requirements that are outside their cultural scope. Business people can understand in a language-friendly way the code being produced by the developers.

VOLERE introduced agile stories. Writing agile stories to explain the real business problem helps connect the business requirements of the problem to the **functional** and **non-functional** requirements.

- **Functional Requirements:** are the fundamental parts of the product. They describe what the product has to do and what actions must be done, i.g. a customer can download an e-book from the software.
- **Non-functional Requirements:** are the qualities of the software that the software can have. Different types of qualities can be introduced such as: performance, usability, cost, and security, i.g. downloading this e-book must be fast and reliable.

VOLERE has its original template called *SHELL*. This template helps with the creation of atomic requirements, which are requirements that can be tested, traceable, and well detailed to fit all the requested needs.

2.3.1 *VOLERE* - SHELL Template

The *SHELL* template is a guide to writing atomic requirements more efficiently [42]. Figure 2.14 shows the *SHELL* template also known as a *snow card*. It consists of a number of attributes that when combined together create an atomic requirement. Atomic requirements are the first level of a requirement; they identify what the solution should be doing.

Each atomic requirement is linked to different types of stakeholders. It ranges from the business person to a developer, all the way to the maintainer when the software is launched. Atomic requirements are grouped in a high-level grouping technique. Some important high-level grouping categories are:

- Business Use Cases (BUCs) and Product Use Cases (PUCs)
- Work Scope and Product Scope

- Components and Sub Components

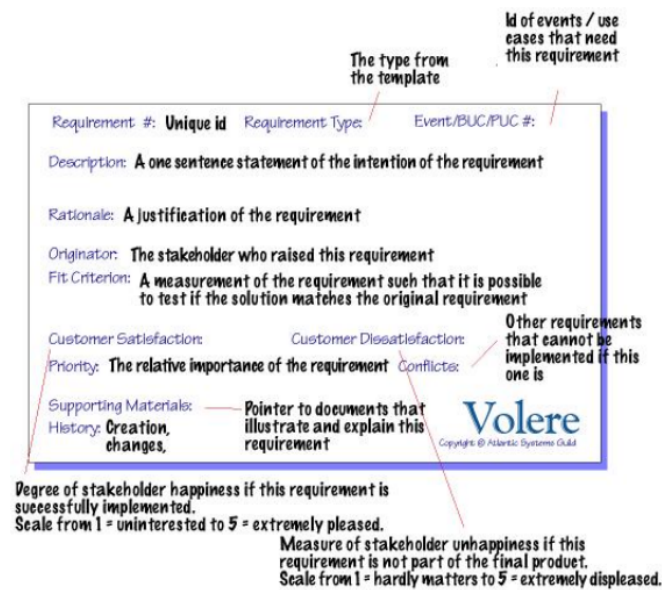


Figure 2.14: SHELL Template

The concept of a language that helps communication between stakeholders working on building a specific software is called *VOLERE*. Now, a tool that can implement this language and make use of the *VOLERE* SHELL template is called *ReqView*. People coming from different backgrounds can use this tool to manage the requirements of a software.

2.4 ReqView

Requirement View ReqView is a Requirements Management Tool. This tool structures requirements to give a better understanding, which will lead to an achievable analysis using *VOLERE*'s language principal discussed earlier. ReqView let its users develop software products using the V-model as shown in Figure 2.15. The V-Model requires each step to be verified and validated.

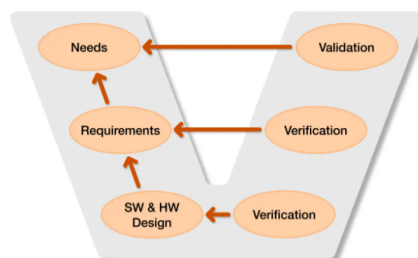


Figure 2.15: V-Model

ReqView can import all types of document. Documents that contain pre-defined requirements from the stakeholders can be verified, checked for risks, and tested, all in one place. Figure 2.16 helps us understand the structure of the documents being imported into ReqView, and how different sections of the requirement analysis is being used.

Volere-REQS: Volere Library Loans Example Requirements

File Edit View Document Project Help

Type: FR Filter

ID	Type	Use Case	Description	Rationale	Fit Criterion
REQS-61	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall accept a Loan Extension Request for the Borrower	Need to know which book loan the borrower wishes to extend	See definition of Extension Borrower Definition
REQS-62	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall find outstanding book loans	Need to know which books are past the loan expiry date	An Outstanding Book Loan is one where the Loan Expiry Date is before Today's Date
REQS-63	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall inform the borrower if the loan extension has been refused along with the reason for refusal	To keep the Borrower informed	See definitions of Refused Loan Extension, Borrower in Terms and Definitions
REQS-64	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall inform the librarian if the loan extension has been refused along with the reason for refusal	To keep the Librarian informed	See definitions of Refused Loan Extension, Librarian in Terms and Definitions
REQS-65	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall record that the extension has been refused	To keep a record that the borrower has been informed	See definitions of Refused Loan Extension in Terms and Definitions
REQS-66	Functional Requirement	NEEDS-88: Determine Extension Eligibility	The product shall record the new loan expiry date	To keep track of extended loans	the new Loan Expiry Date is the Extension Request Date

INFO × NEEDS × REQS ×

Upstream Traceability to Product Use Cases

Requirements Attributes

Figure 2.16: ReqView Template

Traceability reports are generated after the requirements analysis. After reviewing all important procedures of a requirements study, ReqView sums up the work that has been done by generating traceability reports, as shown in Figure 2.17. In this traceability report, the "ID" and "Description" of a requirement are used by the user to check which requirement is being analyzed, each row at a time. A status shows as approved means that a requirement has passed all the verification tests and is now eligible to be used in the production phase. Finally, if the needs of a specific requirement are satisfied, then no more work is needed for that requirement. Traceability reports help stakeholders understand which requirement needs attention and which is ready to be implemented.

ReqView SRS Traceability Report

Satisfied Needs	ID	Description	Status	Verified by Tests
	SRS-20	2 Requirements		
	SRS-21	2.1 External interfaces		
	SRS-22	2.2 Functions		
	SRS-51	2.2.1 File Operations		
	SRS-52	2.2.1.1 Create Document		
NEEDS-58 Document Files Tool supports: • saving project in a file • opening project from a file	SRS-53	The application shall allow users to create a new empty document.	Approved	TESTS-21 Create Requirements Passed
NEEDS-58 Document Files Tool supports: • saving project in a file • opening project from a file	SRS-54	If the current document contains unsaved changes then the application shall allow users to save the changes before closing the document.	Approved	
	SRS-55	2.2.1.2 Open File		

Figure 2.17: Traceability Report

ReqView is built to help the process of managing requirements. Stacking up requirements is a common thing in the development process. Managing requirements is a skill that can be achieved with the help of a tool that creates, analyzes and tests all the possible

requirements found in the software environment. ReqView has pre-defined templates for different types of requirements specifications such as: Software/System/Business Requirements Specifications. ReqView comes with a version control tool to be used when mistakes are made and to backtrack to a correct solution.

To understand how we need to elicit different types of symbols from the end-user, we first need to have a solid background on how to conduct an elicitation study. *Gelicit* used a unique approach to elicit gesture controls from its end-users.

2.5 Gelicit

Gelicit (Gesture elicitation), a cloud platform to elicit 2D (stroke) to 3D (mid-air) gestures from participants based on tests, questionnaires and forms. Gelicit can conduct elicitation studies more efficiently from a cloud platform perspective in just **6** steps. These steps are necessary to gather all the information needed to complete a Gesture Elicitation Study (GES). Gestures can offer an intuitive way to interact with a computer [25]. First, they define a study, where designers create a set of missions for the user to accomplish. Then, they have to conduct the study according to Step 1, where any participant who received an invitation is asked to answer all the intended questions that have been specifically chosen for the study. After gathering all the requirements, a gesture classification procedure is performed based on clustering criteria. Calculations on gesture measurements like the agreement rate, are important to understand how different gestures are being produced. Based on the previous step, the designers conducting the study discuss with the participants the different gestures proposed to reach consensus. Finally, after reaching a consensus, the gestures being chosen are exported to be analyzed with a gesture recognizer. Figure 2.18 gives a summary of the workflow being used in the Gelicit software.

Software	Define	Conduct	Classify	Measure	Discuss	Export
GELICIT	Task set, referents, context of use, criteria,...	Elicited gestures by any registered participant, distributed in time and space	Multiple criteria, vocabulary, and labelling	Agreement rate, frequency,...	Rating, ranking, testing	Consensus set with recognizer

Figure 2.18: Workflow Steps

Gelicit solves a reoccurring constraint in an elicitation study. For example, having to gather all stakeholders in a lab can be a time-consuming process. People are in different locations, have different schedules and priorities, so having to think about how each of them can be involved in the study will make the situation much more complex for researchers. Gelicit is a zero-install software, meaning that services that are being provided can be accessed by anyone. Conducting Gesture Elicitation Studies (GES) using a zero-install approach, can be a solution to the stakeholders' problem. Accessing the platform from anywhere, anytime and using any device, would help the study to be conducted efficiently and with less problems.

Gelicit, the cloud platform software, consists of 3 parts. We start by introducing its **meta-model** (a model of a model) like Figure 2.19, where we understand the structure of how gestures are being elicited. Then, the **6 steps** workflow process for conducting a GES as shown in Figure 2.20. In the last part, we will talk about the **software** itself: Gelicit, shown in Figure 2.21.

2.5.1 Meta-model

Meta-modeling is the development of models, constraints, and rules. The main goal of a GES is to elicit gestures. A gesture set is a model that has its own set of classes relying on having a: vocabulary, gesture class, and gesture sample. We can see in Figure 2.19 that the vocabulary encloses all gestures that are being elicited according to the purpose of the study being conducted. The gesture class regardless of its contexts, collects all the gestures samples based on users performing a set of tasks required from them. Any user who is registered by adding their demographic information has access rights to have their gesture samples analyzed. Each participant has its own identifiers linked to his or her gesture samples, such as his age, level of expertise, and profession. Subjective evaluation is required from users who took part in the GES measure on a scale from 1 to 10, 10 being the best possible gesture for the referent [51, 53].



Figure 2.19: Gelicit's Meta-model

2.5.2 The 6 Steps Process

Define

Before conducting any study, researchers must define the overall process of their study. This will help stakeholders and the end user to be aware of their contribution to the study.

Conduct

The study defined earlier must be conducted to start getting the results. An important feature when conducting a GES is if configuration files are being reused or not. Environments can change, Gelicit supports the use of multiple contexts when eliciting gestures from its end-users.

Classify

This step requires knowledge and can be different for each stakeholder. Stakeholders can determine which classification criteria are beneficial based on certain requirements that have been elicited from the end-users. Different criteria can be expressed by stakeholders, each having their own, and then discussing to find a way to reach consensus to use the best among all the criteria.

Measure

Gelicit has a goal. It uses measurements such as the agreement scores [51] and rates [51] to verify end-user agreement and disagreement rates based on elicited gestures.

Discussion

This aims at performing discussions between two parties involved in the study: stakeholders and participants. It helps to discuss the results that have been obtained, analyze them, and fix the important parts that have been negotiated on. This will help designers have a better understanding of their results and how they can change the process according to the solution proposed when reaching consensus between the two parties.

Export

When reaching a set of consensus, Gelicit is able to export the elicited results using a recognizer. Gelicit uses a recognizer called *!FTL*, which is a vector-based stroke recognizer [46], so Gelicit is able to capture gestures with different coordinates.

Figure 2.20 shows the 6 steps process based on snapshots that has been taken from the Gelicit software itself.



Figure 2.20: 6 Steps Workflow of GES in Gelicit [30]

2.5.3 The Software

Gelicit is considered the first cloud computing collaborative platform for conducting a gesture elicitation study with stakeholders distributed in time and space [30]. The aforementioned steps were carried out by eight different stakeholders, each having their respective role in this study. The process starts with two designers working hand in hand

in a small and medium size enterprise (SME) and on Moodle groups, respectively. A user interface (UI) designer, that was responsible for designing the gestures for each task. Two developers, that created the user interface designed by the designers using the *MEAN* stack: *M*ongoDB, *E*xpressJS, *A*ngular and *N*odeJS. The platform has been deployed on *Heroku*¹ a Platform as a Service (PaaS) that can run applications on the cloud. Then we have an experimenter, used in a GES, for conducting and assembling tasks. End-users have a vital role in this study in proposing their gestures based on a series of tasks that are requested from them. Lastly, we have a moderator, who works alongside the author, he has a responsibility to supervise the GES process being done by the focus groups. Problems can occur with any of the described stakeholders, so the roles can be interchanged to be united and resolve any problems along the way. Gelicit implemented three stroke-gesture recognizers: \$1 [6], \$P [48], and !FTL [46]. These recognizers can be activated in the canvas where users can draw their gestures. Then, the user chooses which recognizer they want to use and presses the "recognize" button to begin the process. The process begins by taking all the drawn gestures, storing them, and do a comparison to a pre-defined samples of gestures.

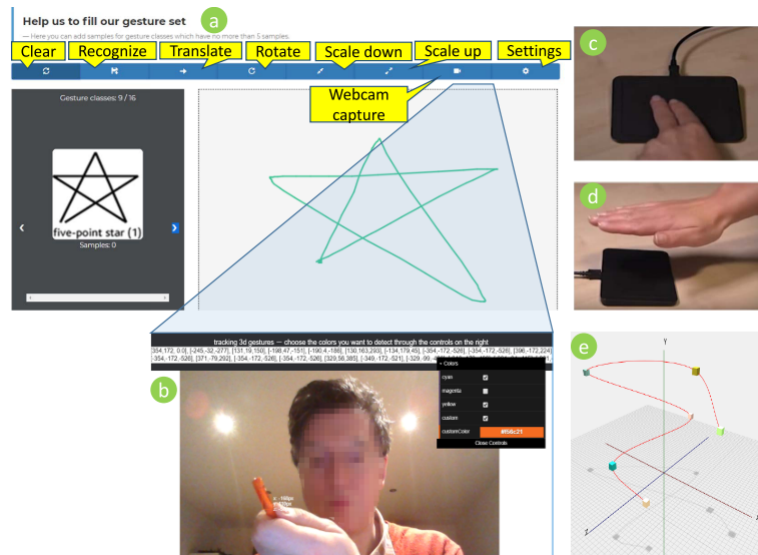


Figure 2.21: Gesture acquisition: (a) main area, (b) 2D1/2 object tracking, (c) touch + (d) air gesture, (e) 3D gesture [30]

Many studies can be conducted using different techniques. When doing a GES, understanding the differences between the techniques (see figure 2.21) is an important process when eliciting gesture controls. The two different techniques are: **direct acquisition** and **indirect acquisition**:

- **Direct Acquisition:** A gesture is acquired as it is from the device that is able to

¹<https://heroku.com>

capture gestures and analyze them automatically. The output would be the same as the user's input.

- **Indirect Acquisition:** A gesture is acquired using the camera itself. No other devices are needed in this technique. This would require the implementation of machine learning gesture recognition software that would assist the process of acquiring gesture controls.

Many software like GestureWiz [41], uses both techniques described earlier. On the other hand, Gelicit focuses on the **direct acquisition** technique. The choice of the technique depends mainly on what gestures can be analyzed using gesture recognition software. In order to avoid the limitation of just having predefined sets of gestures, Gelicit was open to elicit a wide range of gestures controls from the end-user. Gelicit's main framework resides on HTML5, this comes with the limitation that sophisticated gestures cannot be captured.

Gelicit used face-to-face and remote meeting techniques to review their end-user proposals. Having a crowd of end users would make this task inefficient, since researchers would not be able to get feedback and have discussions with all participants involved in their study, especially if hundreds of participants were involved. Figure 2.22 shows the techniques used by Gelicit to elicit gesture proposals. When conducting a study having a large number of participants, where each one of them is able to propose multiple gestures, labeling and categorizing manually each proposal is a time consuming process. Automatic labeling of names for gesture controls would help the process of classifying elicited gestures. This particularly, would have its own study by testing different machine learning algorithms, and see which one fits best based on the gestures given as inputs. These algorithms will automatically categorize and recognize the gestures that are shown.

Software	Devices and modalities	Input format	Output format	Recognizers
GELICIT	2D in a HTML5 canvas by pen, mouse, finger, trackpad, stylus, 2D1/2 by object tracking through webcam, and 3D air+touch gestures	Some types of 2D, 2D1/2, and 3D gestures	Raw data captured by device: (x, y, z, w, t)	\$1, \$P, !FTL,...

Figure 2.22: Gelicit Technical Parameters [30]

2.5.4 Model-Driven Engineering (MDE)

Figure 2.23 shows the different levels of what a model-based engineering process looks like. At the lowest level, we have Model-Driven Architecture (MDA), which is a model-focused approach that focuses on creating an architecture for the software using a subset of UML models. Important models shall be created at the MDA level. We have the Computation Independent Model (CIM) which is also called domain model, here we will find important domain abstractions that are useful in a software. Platform Independent Model (PIM) is described using UML models that shows how the software responds to internal and external events, PIM models software operations without focusing on the implementation. The last model in action is the Platform Specific Model (PSM), and different layers of that model are present, where each layer adds platform-specific details to the PIM.

MDA is encapsulated by Model-Driven Development MDD, which shows how to create software using the architecture described in MDA. As a last point, we need to consider the most important level, which is the Model-Driven Engineering (MDE), which is the whole software being built based on the modeling described at the architecture level.

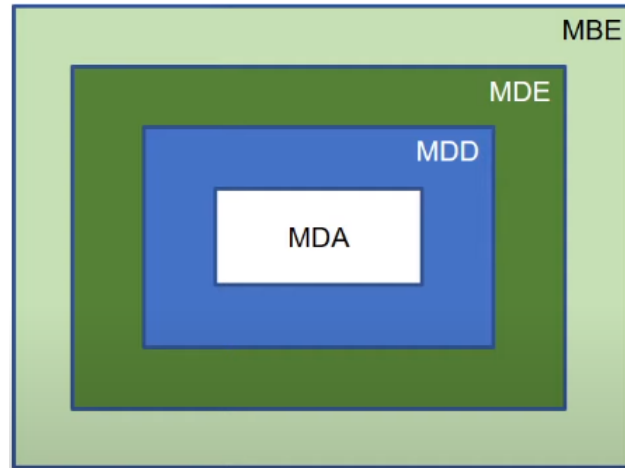


Figure 2.23: Model-Based Engineering

Model-driven engineering (MDE) over the last decade has emerged as a successful and a widely used approach for developing software softwares where models act as the key development artifact [21]. Model-driven engineering (MDE) is an iterative and incremental software development process [24]. Model-driven engineering (MDE) is a subfield in software engineering, and it is used as a way to develop software. The main concept of MDE is the possibility to transform models into code, in other terms, generate a complete software from models. MDE is focused on building domain-specific frameworks, where these frameworks help build software that follows the terminology of a domain. Models are the main outputs of the development process, rather than having programs. The models automatically generate the program by translating the model design into code. Abstraction is an important part when designing these models, it helps engineers not to be concerned with the all the programming language details being used. By generating code automatically, we can adapt software to new platforms in a cheaper way. One of the major advantages of Model-Driven Engineering is the possibility to early assess crucial system properties, in order to identify issues that are easier and cheaper to solve at design level than at code level [31]. So, the main goal of MDE is neglecting the use of knowledge of programming languages by creating models, but when building a software using code, the main benefit is developing the logic and algorithms behind each action. Using the MDE approach, we can describe algorithms using executable (xUML), which is a combination of graphical notation and executable timing rules. xUML is based on an Action Language for Foundation UML (ALF), which is the standard action language that defines operations performed on model elements.

2.5.5 ReDSeeDS

ReDSeeDS is a tool that offers a full Model-Driven Engineering (MDE) process, it allows to have working code that can be generated using use cases, user scenarios and domain models as shown in figure 2.24. It uses familiar concepts such as use cases, scenarios, and domain models to generate detailed design models and code. Additionally, ReDSeeDS

supports searching through requirements models to find similar use cases and domain elements, allowing for reuse of complete solutions to specific problems. The software allows users to edit their requirement models and transform them into UML models and Java code. The framework includes a repository for reusable cases, which are complete sets of related technical artifacts that can be reused in new problem descriptions. ReDSeeDS is built using the concept of Requirements Specification Language (RSL). Requirements specification for an information system is important for several reasons: It serves as a means of communication between the user and the systems developed [45]. RSL is a formal way of expressing the required features and behavior of a software. It usually includes a set of keywords, symbols, and notations that allow for precise communication of requirements among different stakeholders, such as developers, testers, and project managers. RSL consists of use cases, user scenarios, actors, and domain elements. ReDSeeDS has an editor where developers can turn RSL models into pure Java code that can be compiled and executed. A good requirements specification language should be able to capture both functional and non-functional requirements and support traceability and verification. The main strength point of ReDSeeDS is how efficient it is at transforming requirements into readable code. Requirements models such as UML diagrams are transformed into Java code which is the main programming language used in this tool. The code follows the Model-View-Presenter (MVP) approach, where the logic is implemented as the Presenter, all the user interface code is shown in the View, and the domain logic code is found in the Model. With a simple click of a button, ReDSeeDS automatically produces UML models including its code by going through the Model-Driven process. After generating the required code, manual intervention is needed to add the necessary algorithms for the software being built. This approach will facilitate the creation of a software development community focused on the act of **re-usability**.

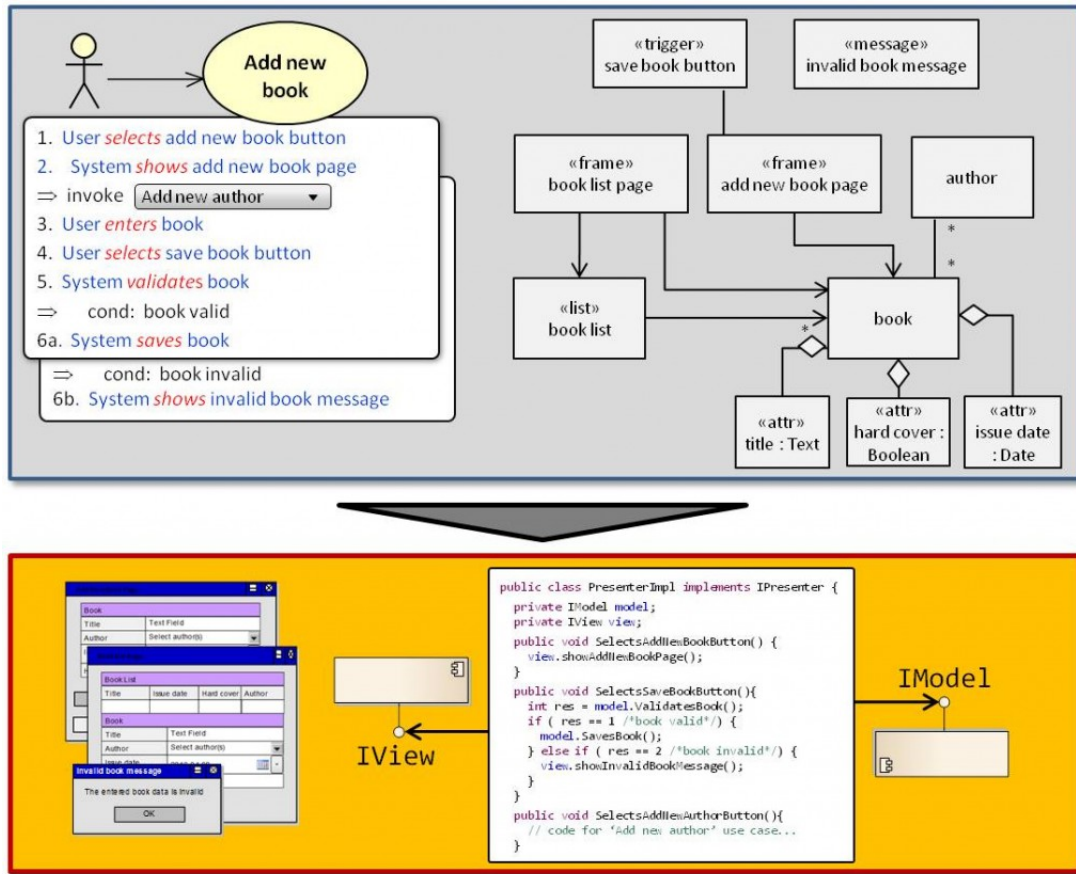


Figure 2.24: Overview of the ReDSeeDS Tool

2.5.6 Other Tools: SnappView

Giving attention to details is crucial when designing a software. This is when a tool like SnappView helps with the design process of an application. [SnappView](#) is an open source software development kit that facilitates end-user review of graphical user interfaces for mobile applications [36]. Figure 2.25 shows an example of the tool in use. We can see that all that has been added are in red, where designers added their comments and the way the user interface needs to change to meet certain requirements. SnappView is a useful tool, especially when different stakeholders need to communicate design issues; it helps them visualize the problem and fix it by adding instructions on the design itself.

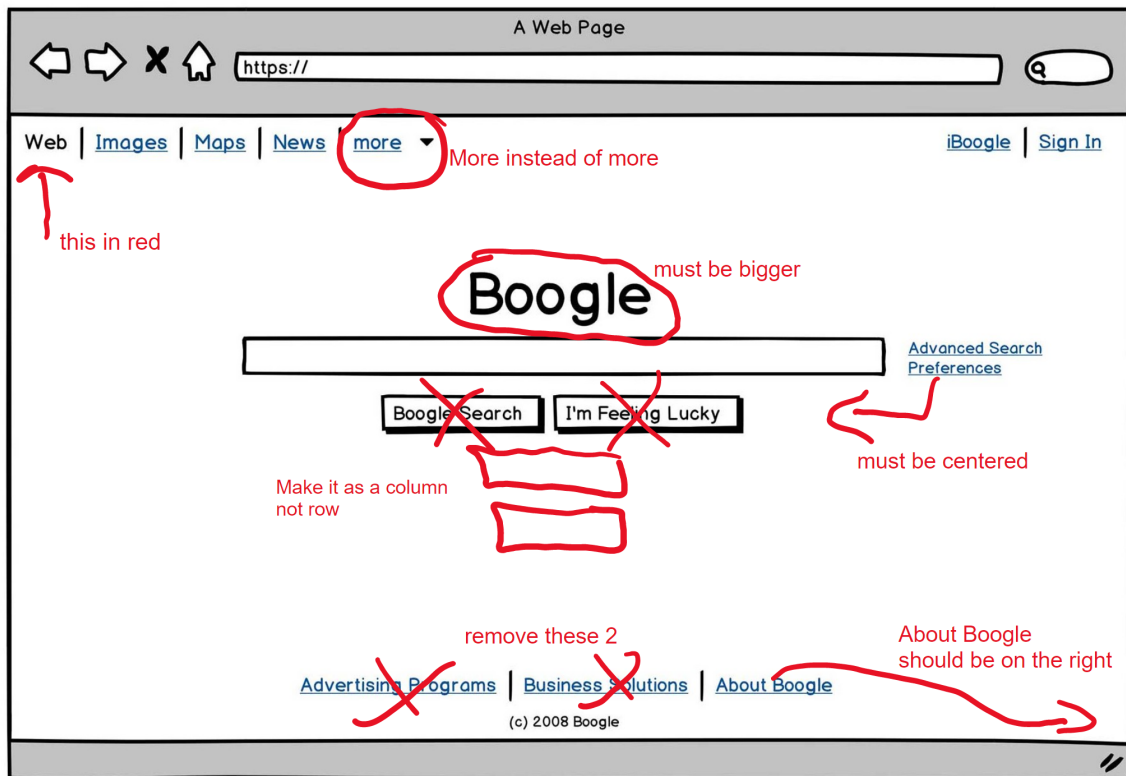


Figure 2.25: SnappView Screenshot Example

2.6 Limitations of Existing Tools

It is important to note that there isn't any tool without a specific set of limitations. The tools we have introduced earlier in this chapter come with their own constraints. These can range from lack of certain features to issues of efficiency and flexibility. We aim to present some of these limitations to better understand potential areas for improvement. Some of these limitations will not be present in *Crowditivity*, our platform, and we will discuss its development and features in detail in the upcoming chapters.

1. **Crowdlcit:** collecting spam answers is a drawback of the Crowdlcit approach [4]. In our platform we will tackle this problem by requiring legit users with real email addresses to effectively use our platform's features. Another limitation of this tool is that it might not be well suited to designing interactions for novel technologies that require users to imagine unfamiliar environments, for example, virtual reality [4].
2. **ReDSeeDS:** its user interface, for instance, may not be as intuitive as desired, presenting navigation challenges especially for non-technical users. This lack of user-friendliness may pose a steep learning curve. Additionally, ReDSeeDS lack the integration capabilities with other utilized project management tools like *Jira* or *Microsoft Project*. This could limit the seamless information flow and collaboration between team members.
3. **SnappView:** One of the key limitations found in SnappView, is with the annotations that are being drawn on an image, this implies that users can draw annotations without any structure behind them. For instance, in our platform, we implemented

a functionality that allows the annotations being drawn on image canvases to be recognized, this provides extra information to the designers to handle cases where *free-form* annotations can be used with a purpose.

2.7 Conclusion

To conclude this chapter, we will take a look at what was discussed. We saw that when we need to develop software, a methodology is needed to help identify all the necessary development phases; this is where we introduce the software development life cycle (SDLC). We describe how the requirements engineering process is the first step in SDLC, which defines and documents the requirements discussed with the client in a clear way. This process has different purposes, so it is important to understand how we can use it in a project that requires attention to details such as user interfaces. We also saw how an end-user elicitation study is conducted and how the Crowdesign engine which consists of Crowdlcit alongside Crowdsensus can help researchers elicit requirements for their studies. Then, we mentioned how the *VOLERE* language and its SHELL template are a way to assist the communication between different stakeholders to understand requirements. We also discussed how we can implement the *VOLERE* concept using ReqView which is a tool that manages different types of requirements. To have a better understanding of what types of elicitation studies can exist in the real world, we analyzed the Gelicit platform that was created to gather different types of gestures based on specific requirements while doing the 6 steps process. Moving on to ReDSeeDS which a tool that can transform use cases and domain models automatically into working Java code, it is based on the Model-Driven Engineering view which is an important part in the Requirements Engineering process. Other tools such as SnappView can be used to define requirements in a clear and concise way, where a user can identify them by modifying the screenshot of a software. Finally, we have seen some of the limitations of these tools.

Chapter 3

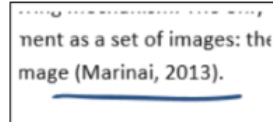
Eliciting and Identifying Annotation Types

This chapter will focus on creating effective *annotation types*, where an *annotation type* is a class of annotations that shares similar properties [19]. To do that, we will conduct a *crowd elicitation study*, which involves gathering different types from a crowd of participants. These participants will perform a series of tasks using tools such as a mouse, trackpad, or their own fingers to create annotations on digital canvases. To choose the best type of annotation, we will conduct a survey that aims at identifying various types of annotations by presenting participants with a series of questions that required them to select a certain type. We will discuss the process of analyzing and classifying these annotation types and see how the AGATe [49] tool helped calculate the *agreement rate* of the generated types.

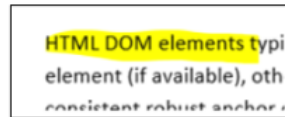
3.1 Definition of Annotations

In user interface design, an *annotation* is a mark made on a design mock-up to give instructions about specific design elements that need to be changed. For example, in situations involving text, the term annotation comprises several methods, including underlining and highlighting text and writing additional comments in the margin [23]. These activities are shown to stimulate critical thinking in a process that can be called active reading [2]. In our case, annotations serve as a communication technique between designers, developers, and other stakeholders working on software development. One important role of annotations is to act as signposts to important information [13], and providing such information helps ensure that the software meets the requirements and that all stakeholders have a clear understanding of its design. No matter the form or purposes of the annotations, the benefits are immediately clear [1]. A key goal in designing an interface is to minimize the amount of effort required for a new user to become familiar with the interface. Figures 3.1 and 3.2 are examples of the different types of annotation that a user could annotate on a **text-based image**.

Single Line Annotations

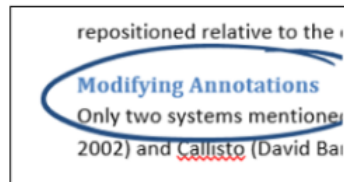


Underline

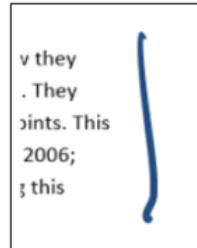


Highlight

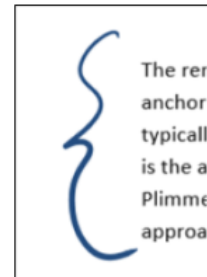
Multiple Line Annotations



Enclosure

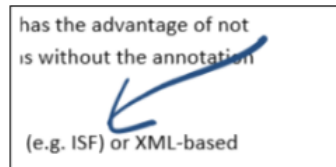


Margin Bar



Brace

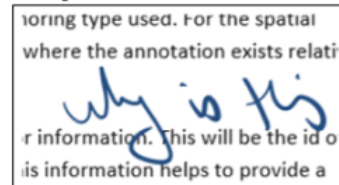
Connector Annotations



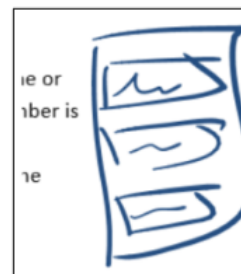
Callout/arrow

Figure 3.1: Example of annotation types [44]

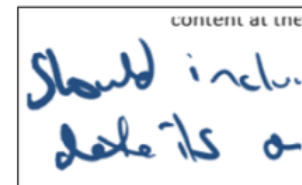
Complex Annotations



Text/symbol

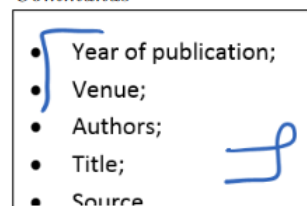


Drawing



Marginalia

Commands



Command

Figure 3.2: Continued [44]

3.2 Elicitation Study of Annotation Types

Involving customers in the development process is a must, not only while generating requirements, but also considering that requirements change quickly and must be continuously validated [14]. Having many users participate in an elicitation study would give us

the ability to better understand what they need to reach a good outcome. Our aim was to find a platform that could enable us to generate online surveys while also allowing space for users to annotate their ideas. We found Forms.app¹, an online form builder that offered the feature that allows users to annotate their symbols on canvases. After deploying our questionnaire, we conducted a thorough data analysis using the analytics section of "Forms.app", which provided us with statistical tools. To have people participate in our elicitation study, we first introduced our research purpose in the introduction section, then defined the tasks needed to achieve our study goals. Moreover, we asked for demographic information to guarantee the research findings' credibility. In the following section, we are going to see the questions that were asked, the various demographic information of the participants that were involved, and how we classified the different annotation types based on the collected data. An example of this study survey can be found in section [Forms.app](#).

3.2.1 Questions Asked

- **Introduction:** "You are invited to participate in our survey related to the evaluation of annotation types. Your participation in this study is completely voluntary. There are no foreseeable risks associated with this project. It is very important for us to know your opinion on certain tasks. Your survey responses will be strictly confidential, and data from this research will be reported only in the aggregate. Your information will be encrypted and remain confidential. Thank you very much for your time and support. Please start the survey now by clicking on the Start button below. "
- **Demographic Information:**
 1. What is your age?
 2. What is your gender?
 3. I use a smartphone frequently on a scale from 1 to 7?
 4. I use a computer frequently on a scale from 1 to 7?
 5. I use a tablet frequently on a scale from 1 to 7?
- **List of Questions:** Figure 3.3 shows an example on how the user is prompted with a question and a canvas to draw their annotation symbol:

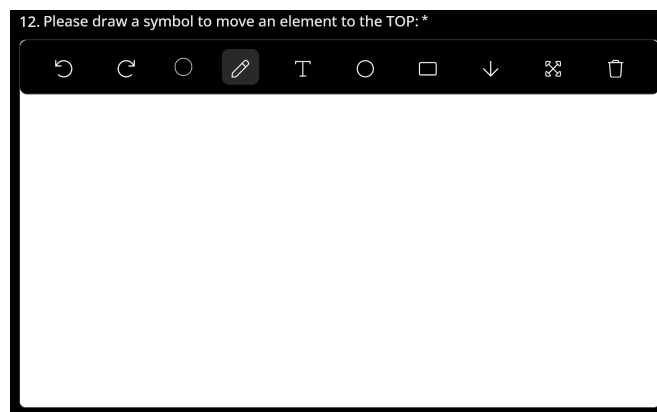


Figure 3.3: Example of a question on forms.app

¹<https://forms.app/>

1. "Please draw a symbol for selecting an element", this task asks the participants to create a symbol that can be used to indicate when an element has been selected. This symbol could take various forms, such as a rectangle, a circle, or a triangle.
2. "Please draw a symbol for deleting an element", this task asks the participants to create a symbol that represents the action of deleting or removing an element from an interface. The symbol could be a cross, an X, or any other symbol that has the idea of deleting an element.
3. "Please draw a symbol for adding an element", this task asks the participants to create a symbol that represents the action of adding or inserting an element into an interface. The symbol could be a plus sign, a downward arrow, or any other symbol that could mean adding an element.
4. "Please draw a symbol for showing attention to an element", this task asks the participants to create a symbol that can be used to draw attention to a particular element on an interface. This symbol could take various forms, such as an exclamation mark, a blinking icon, or an effect that draws the user's eye that this element is important.
5. "Please draw a symbol to enlarge/zoomIn on an element", this task asks the participants to create a symbol that represents the action of zooming in or enlarging an element on an interface. The symbol could be a magnifying glass, a plus sign, or any other cue that could have the idea of enlargement.
6. "Please draw a symbol to shrink/zoomOut on an element", this task asks the participants to create a symbol that represents the action of zooming in or shrinking an element on an interface. The symbol could be a magnifying glass, a minus sign, or any other cue that could have the idea of shrinking an element.
7. "Please draw a symbol to move an element to the TOP", this task asks the participants to create a symbol that represents the action of moving an element to the top of an interface. The symbol could be an upward arrow, a stack of papers, or any other symbol that could lead to move an element upward.
8. "Please draw a symbol to move an element to the BOTTOM", this task asks the participants to create a symbol that represents the action of moving an element to the bottom of an interface. The symbol could be a downward arrow, a pile of papers, or any other symbol that could lead to move an element downward.
9. "Please draw a symbol to move an element to the LEFT", this task asks the participants to create a symbol that represents the action of moving an element to the left side of an interface. The symbol could be a leftward arrow, a stack of papers shifted to the left, or any other symbol that could lead to move an element to the left.
10. "Please draw a symbol to move an element to the RIGHT", this task asks the participants to create a symbol that represents the action of moving an element to the right side of an interface. The symbol could be a rightward arrow, a stack of papers shifted to the right, or any other symbol that could lead to move an element to the right.
11. "Please draw a symbol to show an element is good", this task asks the participants to create a symbol that can be used to indicate when an element

is considered "good" or acceptable. The symbol could be a checkmark, a thumbs-up, or any other visual cue that communicates the idea of approval.

12. "Please draw a symbol to show an element is bad", this task asks the participants to create a symbol that can be used to indicate when an element is considered "bad" or unacceptable. The symbol could be an X, a thumbs-down, or any other visual cue that communicates the idea of disapproval.
13. "Please draw a symbol to rotate an element LEFT", this task asks the participants to create a symbol that represents the action of rotating an element counterclockwise. The symbol could be a leftward arrow with a curved tail, a circular arrow that points left, or any other visual cue that communicates the idea of a left rotation.
14. "Please draw a symbol to rotate an element RIGHT", this task asks the participants to create a symbol that represents the action of rotating an element clockwise. The symbol could be a rightward arrow with a curved tail, a circular arrow that points right, or any other visual cue that communicates the idea of a right rotation.
15. "Please draw a symbol to show that this element is required", this task asks the participants to create a symbol that can be used to indicate when an element is required on an interface. The symbol could be an asterisk, a star, or any other visual cue that communicates the idea of mandatory element.
16. "Please draw a line", this task asks the participants to draw a simple straight line.
17. "Please draw to show confusion on an element", this task asks the participants to create a symbol that can be used to indicate when a user is unsure about a particular element on an interface. The symbol could be a question mark, a thinking bubble, or any other visual cue that communicates the idea of confusion.
18. "Please draw to show that this element can be OPTIONAL", this task asks the participants to create a symbol that can be used to indicate when an element is optional on an interface. The symbol could be a grayed-out or light-colored version of the required symbol, a question mark, or any other visual cue that communicates the idea of an optional element.

3.2.2 Participants' Information

We sent our survey around a group of people that were kind enough to take part in our elicitation. Based on the series of questions that we saw earlier, we gathered a good understanding on the participants' background, where each one had their own unique perspective when it comes to creating an annotation. Table 3.1 provides us with information about the people who participated in our inquiry, their age, gender, and how frequently they use a "Smartphone", "Computer", and "Tablet" (based on a 7-point Likert scale [29] ranging from 1=strongly disagree to 7=strongly agree).

ID	Age	Gender	Smartphone	Computer	Tablet
1	23	Male	7	7	2
2	26	Male	7	7	1
3	19	Female	7	4	7
4	23	Male	7	7	2
5	21	Male	7	7	6
6	22	Male	7	7	1
7	24	Male	5	7	1
8	23	Female	7	7	7
9	18	Male	7	7	1
10	57	Male	5	7	4
11	29	Male	7	7	2
12	46	Male	7	7	7
13	34	Female	7	7	2
14	47	Female	7	7	4
15	38	Female	7	7	3
16	22	Female	7	7	1
17	34	Male	7	7	7
18	32	Female	6	7	5
19	27	Female	7	5	1
20	30	Male	7	7	3
21	19	Male	7	6	5
22	23	Male	7	7	1
23	24	Female	7	6	1
24	25	Female	7	7	7
25	26	Female	7	7	1
26	36	Female	7	7	3

Table 3.1: Participant Demographics in the Elicitation Study: Sample size of 26 Participants

Furthermore, figure 3.4 shows the pyramid chart that represents the diversity of male and female participants who took part in our elicitation. This chart shows that our sample was evenly split between male and female participants, this provides a good representation of both genders' perspectives on the annotation types that were elicited.

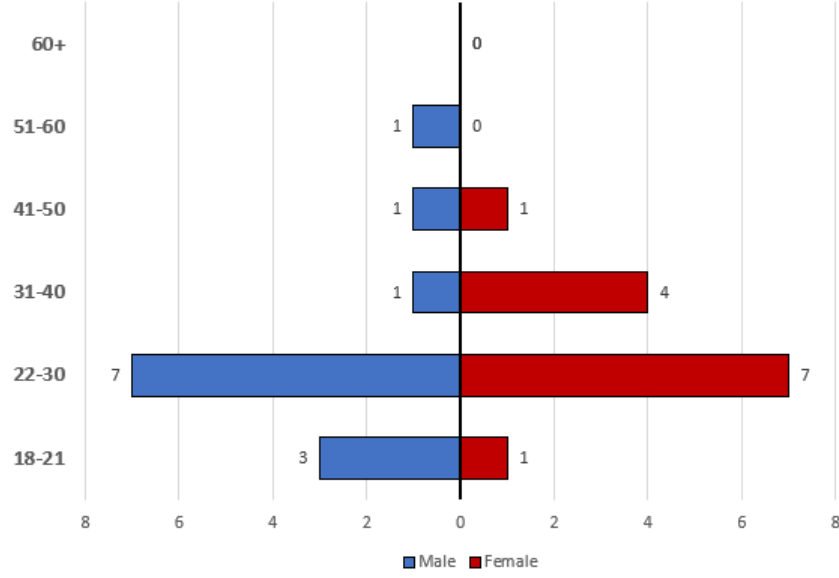


Figure 3.4: Age Distribution of Participants in the Elicitation of Annotation Types Study

Moreover, to have a better understanding on our participants' technology usage preferences, we created a Likert chart (see Figure 3.5) to visualize the agreement scaling of the participants involved. This chart allowed us to examine the percentage of agreement or disagreement among participants on a range of statements related to their use of certain devices and technology. On the basis of this graph, we can see that the participants involved in our elicitation used both computers and smartphones frequently.

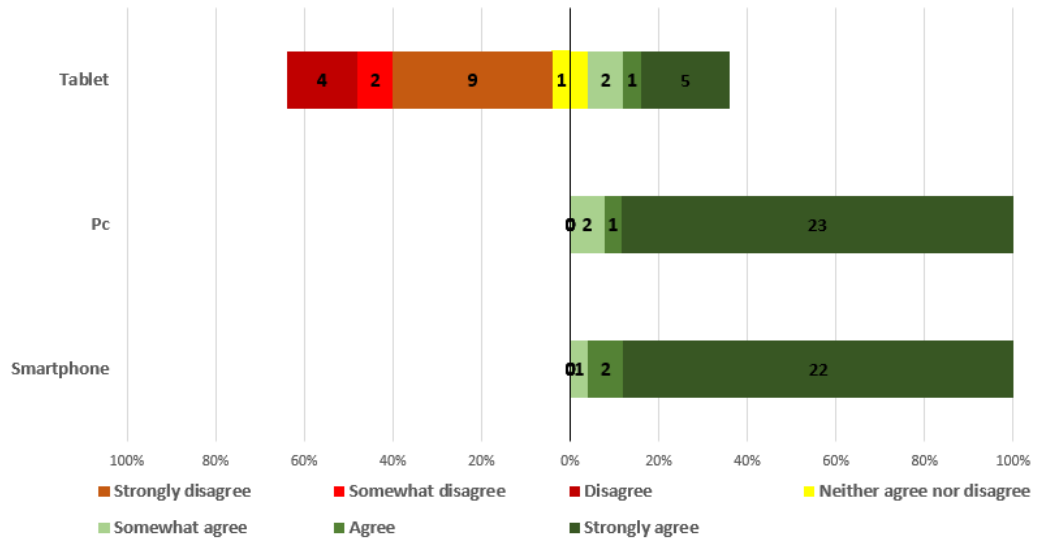


Figure 3.5: Likert Chart on Agreement Scaling - Eliciting Annotation Types

Having participants of different ages, genders and levels of experience with technology, we can ensure that the findings are more representative to a wider group of people. Next, we will do a *classification* on the data and introduce *AGATe* [49], a tool that was used to calculate the consensus among the participants.

3.2.3 Classification of Data

Once the study is closed, we need to classify the results by checking each annotation elicited for each task performed. By doing so, we can calculate agreement rates and obtain meaningful results. To be able to do a proper classification on the data, we clustered the annotation types into 6 groups where similar types of annotations grouped together:

- **Shape:** this group has annotations related to the shapes drawn on an image. Annotation categories like "Line" is used as a basic shape, "Add" and "Delete" are used to add or remove elements, respectively, and "Select" is used to select an element.
- **Position:** this group includes annotations related to the movement of an element on an image. "Move left," "Move right," "Move down," and "Move up" are used to move an element in different directions.
- **Rotation:** this group includes annotations related to the rotation of elements on an image. "Rotate left" and "Rotate right" are used to rotate an element left and right, respectively.
- **Zoom:** This group includes annotations related to the size of an element. "Enlarge/Zoom in" and "Shrink/Zoom out" are used to increase or decrease the size of an element, respectively.
- **Feedback:** this group includes annotations related to the feedback given by the users regarding a specific element. "Confused" is used when a participant is uncertain about what a specific element is used for, while "Good" and "Bad" are used to indicate whether the element is correctly designed or not. "Attention" is used to indicate that the element shown on the canvas is important and requires some kind of attention.
- **Requirement:** this group includes annotations related to the existence of an element in an image. "Required" is used to indicate that an element is mandatory, while "Optional" is used to indicate that it is not necessary but can be left as is.

After we have identified these groups, we created a table (see figure 3.6) to showcase which annotations belong to which group:

Annotation	Annotation Category	Description	Classification
	1: Line	Annotate an element to strikethrough or underline	Shape
	2: Move left	Annotate to move the an element to the left	Position
	3: Move right	Annotate to move the an element to the right	Position
	4: Add	Annotate to add an element	Shape
	5: Required	Annotate to make this element required	Requirement
	6: Move down	Annotate to move the an element to the bottom	Position
	7: Rotate left	Annotate to rotate an element left	Rotation
	8: Rotate right	Annotate to rotate an element right	Rotation
	9: Move up	Annotate to move the an element to the top	Position
	10: Confused	Annotate to make an element appear unclear	Feedback
	11: Good	Annotate that this element is positively designed	Feedback
	12: Bad	Annotate that this element is negatively designed	Feedback
	13: Delete	Annotate to remove or delete an element	Shape
	14: Attention	Draw attention to an elementt	Feedback
	15: Enlarge/Zoom in	Annotate to make an element's size bigger	Zoom
	16: Select	Annotate to select an element	Shape
	17: Shrink/Zoom out	Annotate to make an element's size smaller	Zoom
	18: Optional	Annotate to make this element optional	Requirement

Figure 3.6: Classification of annotation types

The agreement rate is a score from 0 to 1, which is defined by Wobbrock et al. as “the degree of consensus among participants” [52]. We have gathered data from the 26 participants that were involved in our study, where each participant created 18 different annotations. We grouped the annotations created by participants in our study and can be found in the Appendix section B). Now, to be able to analyze the results on *AGATe* [49], we need to group all the referents (explained here 2.2.1) with all the different annotations that were created. In total, 26 participants x 18 types = **468** elicited annotations. So, since some of the elicited annotations were similar, we created 74 groups, and to better understand how these groups were created, we will look at the example in figure 3.7. When participants were prompted to answer the question "draw a symbol for showing attention to an element", we can see that their annotations were not all the same, so we needed to group them based on their similarity. For this example, we were able to define nine different classification groups, and each group is numbered next to each annotation created by a single participant. We noticed that some participants used the same type of annotation in different scenarios, for example, we have group "11" and group "12" that were also used to show that an element is "required" and show "confusion", respectively, but some participants interpreted them as an annotation that shows "attention" as well. The grouping of annotation types was done by combining all the similar types together and assign them a number that defines each group. Group 6 was created by grouping all the annotations that have a vertical line and a dot without anything else around it, and if we compare group 6 to group 8, we can see that participants chose to add a triangle around that vertical line and the dot, this emphasizes more on the importance of that element. Groups 13 and 14, were identified as groups that only have a single annotation, which means that they were not used in the same question again or in different annotation types, so we considered them as **standalone** groups that the user interpreted to show "attention". After classifying all the 18 annotation types, we will compare each participant's annotation to the group that would be the best fit. This process was carried out using the 74 groups on all results of the 26 participants.

8 	6 	6 	7 	6 	12
8 	6 	6 	6 	7 	
8 	9 	8 	11 	8 	
6 	7 	6 	7 	6 	
14 	6 	10 	13 	7 	

Figure 3.7: Elicited Results for the Annotation of Type "Attention"

The results of this process can be shown in Table 3.2. The format of the table must be respected in order for the *AGATe* [49] tool to correctly interpret the data. The column called "referent" in our case represents the annotation types, and columns p1 to p26 (Person1 to Person26) belong to each participant who was involved in our study. A row in this table represents how each participant interpreted an annotation type by adding the group number that fits best according to the annotation that they created.

referent	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26
Add	1	1	1	1	1	1	1	1	2	3	1	1	1	1	1	1	1	1	1	4	1	1	1	5	1	1
Attention	6	6	7	6	8	6	6	6	7	8	9	10	8	11	8	12	6	13	6	6	6	6	13	14	6	8
Bad	15	16	16	15	16	17	17	16	18	19	20	16	17	16	16	16	17	17	17	16	17	17	16	21	17	16
Confusion	22	23	23	23	24	22	18	23	23	23	23	23	23	23	23	6	23	24	23	25	23	23	23	26	23	23
Delete	16	16	27	16	28	16	16	16	27	29	16	16	16	16	27	16	27	30	16	16	31	16	16	31	16	28
Good	3	3	3	3	3	32	33	3	3	3	34	3	33	3	3	3	33	33	3	3	33	3	35	36	33	3
Line	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
Move Bottom	37	37	37	37	37	37	38	37	39	37	37	39	37	37	37	37	30	37	37	39	38	37	37	38	37	37
Move Left	40	40	40	40	40	40	41	40	41	40	40	41	40	40	40	40	30	40	40	40	40	40	40	40	40	40
Move Right	42	42	42	42	42	42	43	42	43	42	42	43	42	42	42	42	30	42	42	42	42	42	42	42	42	42
Move Top	44	44	44	44	44	44	45	44	46	44	44	47	44	44	44	44	30	44	44	45	40	44	44	40	44	44
Optional	48	22	49	22	50	51	52	53	53	54	55	49	56	23	6	23	56	49	22	57	22	22	49	49	56	49
Required	58	58	58	58	58	58	58	6	58	58	6	58	58	6	58	58	8	58	58	59	58	58	58	60	58	58
Rotate Left	38	38	38	38	62	38	38	38	38	38	63	63	38	38	62	63	30	38	38	38	38	38	38	62	38	38
Rotate Right	62	62	62	62	38	62	62	62	62	64	64	64	62	62	38	64	6	62	62	62	62	62	62	38	62	62
Select	52	30	30	30	42	65	52	52	46	52	44	66	46	3	44	30	30	52	52	30	52	30	52	30	3	30
Zoom-in	67	68	68	68	68	68	30	5	68	44	68	68	68	5	5	69	52	30	67	47	68	68	47	5	5	5
Zoom-out	70	71	71	71	71	30	71	72	71	37	67	30	71	73	72	72	30	30	67	74	71	71	37	72	72	72

Table 3.2: Classification of Elicited Annotation Types

To better understand how we defined each category to classify our data, we added all categories that were elicited in our study. Section C shows the results for each type of annotation where in each part, based on the annotation itself, we defined the classification groups from 1 to 74. Table 3.3, shows the respective annotation types and their group range.

Referent	Classification Groups
Add	1, 2, 3, 4, 5
Attention	7, 8, 9, 10, 13, 14
Bad	15, 16, 17, 19, 20, 21
Confusion	18, 24, 25, 26
Delete	16, 27, 29
Good	32, 33, 34, 35, 36
Line	28
Move Bottom	37, 39
Move Left	31, 40, 41
Move Right	42, 43
Move Top	44, 46
Optional	22, 23, 48, 49, 50, 51, 53, 54, 55, 56, 57
Required	11, 58, 59, 60
Rotate Left	38, 63
Rotate Right	45, 62
Select	30, 52, 65, 66
Zoom-in	47, 67, 68, 69
Zoom-out	12, 70, 71, 72, 73, 74

Table 3.3: Referents and their Classification Groups

3.2.4 AGATe

AGATe [49] (Agreement Analysis Toolkit), is a software tool developed by Jacob O. Wobbrock. "It computes agreement measures and statistical tests for agreement rates" [49]. "The tool is implemented in C# .NET 4.5, and is freely available to download at <http://depts.washington.edu/aimgroup/proj/dollar/agate.html>" [49]. The software is designed to analyze the agreement between multiple participants who annotate the same data. It uses a variety of statistical techniques to determine the level of agreement between them, and it has a user-friendly interface that allowed us to add our data in the format of an Excel sheet. The software computes the agreement rates and provides a visual representations of the results which would help us identify areas of agreements. Based on the results that we elicited, we were able to use this tool to calculate the agreement rates for each annotation type and get an average agreement rate score for all the annotations data. Figure 3.8 shows the mathematical formula to calculate the agreement rate (AR).

$$\mathcal{AR}(r) = \frac{|P|}{|P|-1} \sum_{P_i \subseteq P} \left(\frac{|P_i|}{|P|} \right)^2 - \frac{1}{|P|-1}$$

Figure 3.8: Agreement Rate Formula [49]

The computed agreement rates for each annotation type show different levels of agreement between participants (see Figure 3.9). The highest agreement rate of "1.000" is achieved for the annotation of type "Line", which indicates that all participants provided identical annotations for this type. The average agreement rate across all annotation types

is **0.468**, this indicates that the level of agreement is generally moderate and that there is a certain level of consistency in how participants interpreted these annotations. We can see that confidence intervals (CI) were computed along with the agreement rates. The confidence interval was computed at a 95% level of confidence, meaning that there is a 95% chance that the true agreement rate falls within the calculated interval. It is represented by two values: the lower bound and the upper bound and it is important to note that the CI for some of the agreement rates are wide, indicating that the true agreement rate could be lower or higher than the reported value but ensuring that it would fall in the specified range. For example, in "AR(Add) = .711, CI 95% = [.492,.923]", the agreement rate for the "Add" annotation type is 0.711 and the 95% confidence interval is between 0.489 and 0.923. This means that if the same survey was conducted with different participants, 95% of the computed agreement rates would fall between the range of 0.489 and 0.923.

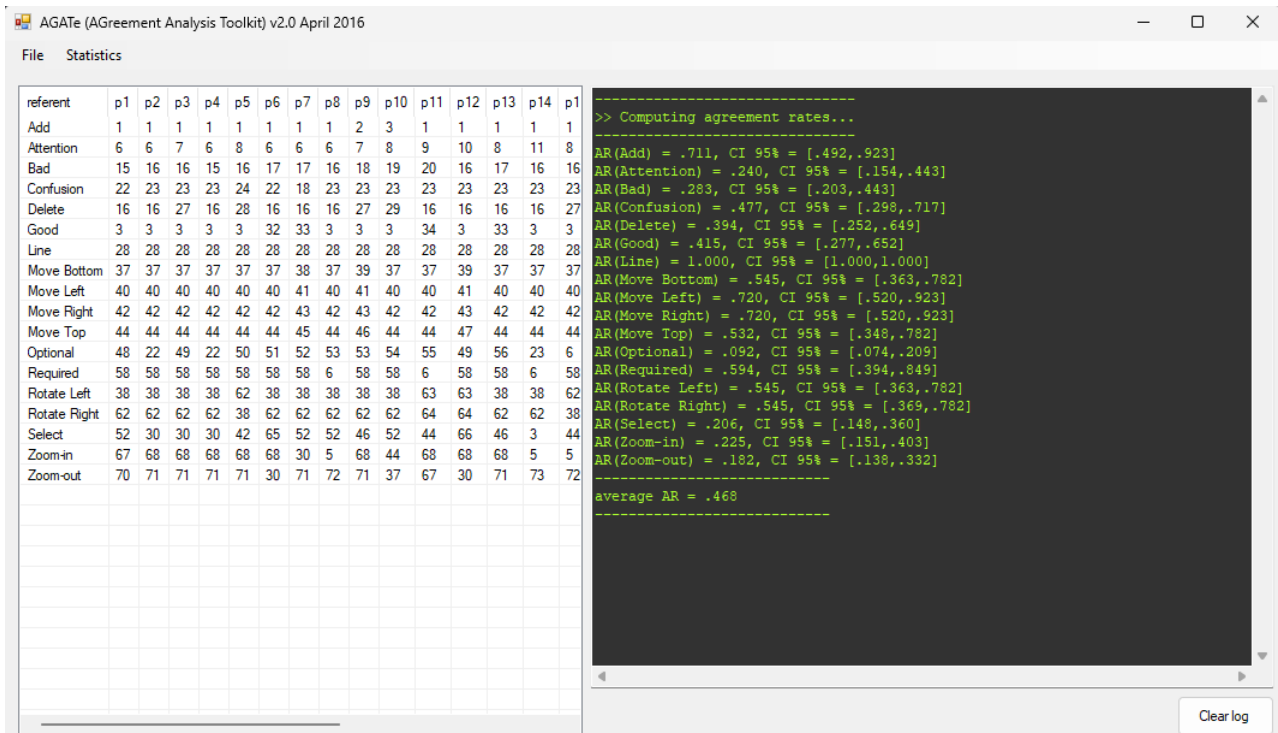


Figure 3.9: Referent Agreement Rates calculated on AGATe [49] using the Classification Data of Table 3.2

Annotation Type	Agreement Rate	CI 95%
Line	1	[1.0001.000]
Move Left	0.72	[.520.923]
Move Right	0.72	[.535.923]
Add	0.711	[.492.923]
Required	0.594	[.397.849]
Move Bottom	0.545	[.372.782]
Rotate Left	0.545	[.372.782]
Rotate Right	0.545	[.372.782]
Move Top	0.532	[.348.782]
Confusion	0.477	[.286.720]
Good	0.415	[.283.655]
Delete	0.394	[.249.649]
Bad	0.283	[.200.449]
Attention	0.24	[.154.440]
Zoom-in	0.225	[.151.400]
Select	0.206	[.148.360]
Zoom-out	0.182	[.138.329]
Optional	0.092	[.074.209]
Average AR	0.468	

Table 3.4: Referent Agreement Rates calculated on *AGATe* [49]

Figure 3.10 shows a better visualization of the agreement rates that have been sorted in decreasing order. We used 5 ranking categories for the levels of agreement, which are: Very high agreement for any referent who has an agreement rate higher than **90%**, in our case only the "Line" is part of this category. High agreement for any referent who has an agreement rate between **70%** to **89%**, like "Move Left, Move Right, and Add". Medium agreement for any referent who has an agreement rate between the average value which is **46.8%** and **69%**, like "Required, Move Bottom, Rotate Left, Rotate Right, and Confusion". Everything below average is considered a low agreement rate like "Good, Delete, Bad, Attention, Zoom-in, Select, and Zoom-out". Regarding the "Optional" annotation type, it has an agreement rate of **<10%** which is considered a very low agreement compared to the other types.

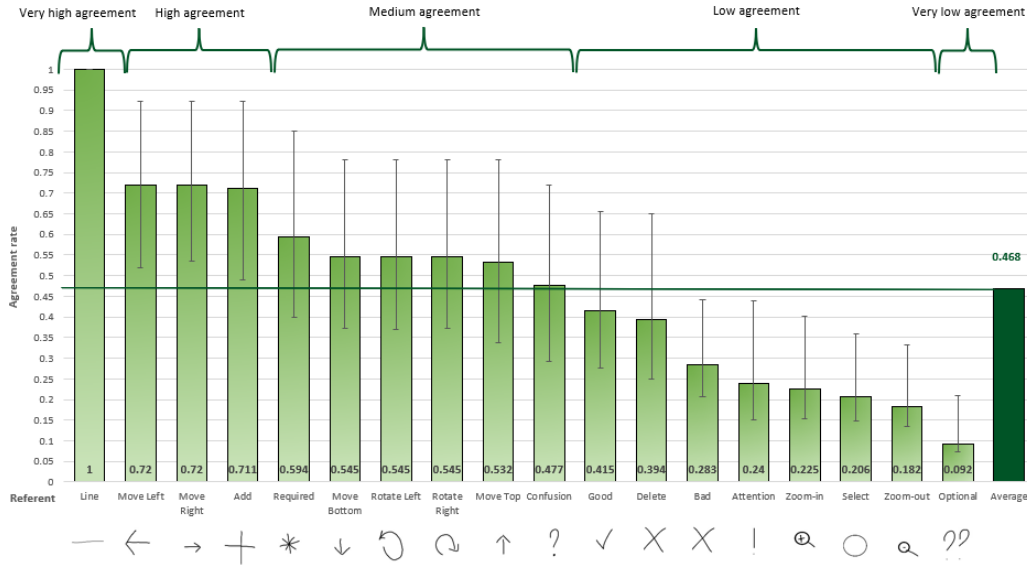


Figure 3.10: Agreement rates that were computed by *AGATe* [49] for referents, which are sorted in decreasing order, with 95% confidence intervals

In the upcoming section, we will explain the identification survey, which is considered an effective way to understand how users interpret annotation types in different situations.

3.3 Identification Study: Annotation Types

Conducting this kind of survey helps establish a common understanding among participants regarding the specific annotation types being identified. By asking participants to identify which annotation they would choose, it provides information on the consistency of the participants' interpretations of each type. This information can be used to improve the way we annotate certain elements, making them more useful in practice, which impacts the decisions that need to be taken regarding the implementation of our annotation and recognition tools, which we will see in detail in the next chapter. The process of creating a survey revolves around the same idea that was discussed earlier in the section related to our elicitation study.

3.3.1 Survey Questions

- **Introduction:** "Hello, you are invited to participate in our survey related to the evaluation of annotation types. Your participation in this study is completely voluntary. There are no foreseeable risks associated with this project. It is very important for us to know your opinion on certain tasks. Your survey responses will be strictly confidential and the data from this research will be reported only in the aggregate. Your information will be encoded and will remain confidential. Thank you very much for your time and support. Please start the survey now by clicking on the Start button below."

- **Demographic Information:**

1. What is your age?
2. What is your gender?
3. I use a smartphone frequently on a scale from 1 to 7?
4. I use a computer frequently on a scale from 1 to 7?
5. I use a tablet frequently on a scale from 1 to 7?

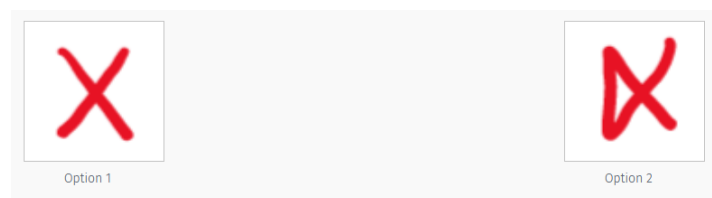
- **List of Questions:** "In the following series of questions, several images of annotation types will be displayed. Your task is to select the element that is the most appropriate according to the instruction describing which type should be used to perform such action."

Based on the classification and the results that we obtained after calculating the agreement rates for different annotation types, we created a survey that is only focused on the types of annotation that were clear enough for the users who participated in our elicitation. In that case, we did not add questions that are related to the following annotations: "Line, Required, Confusion, Optional, and Bad". The main reason was that it was challenging to find the appropriate images that accurately reflected these annotation types, so without carefully choosing the right image for a certain type of annotation, we would cause confusion for the participants. Therefore, the list of questions where participants are requested to choose only **one** of the options is as follows:

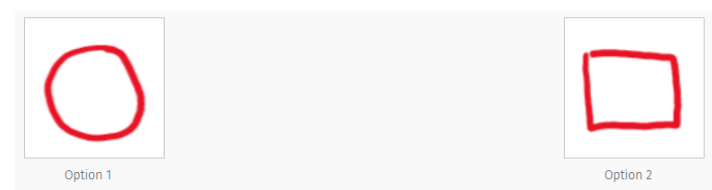
1. "Which annotation would you perform to draw an attention to an element?"



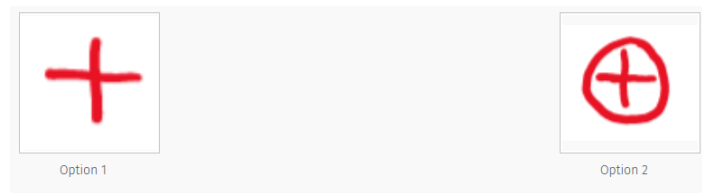
2. "Which annotation would you perform to delete an element?"



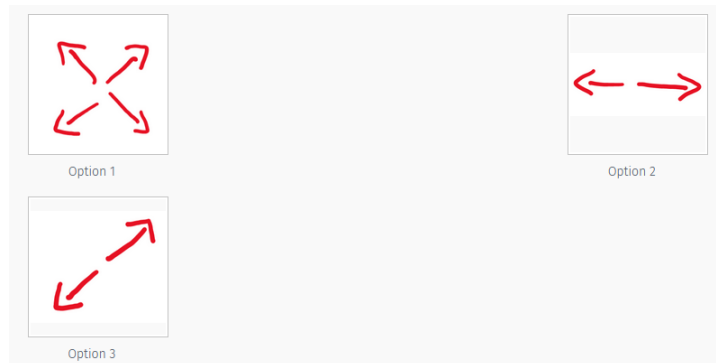
3. "Which annotation would you perform to select an element?"



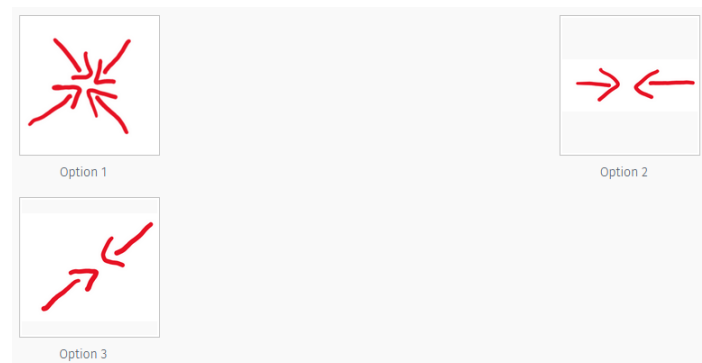
4. "Which annotation would you perform to add an element?"



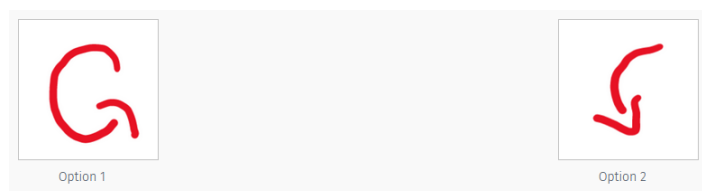
5. "Which annotation would you perform to zoom-out on an element?"



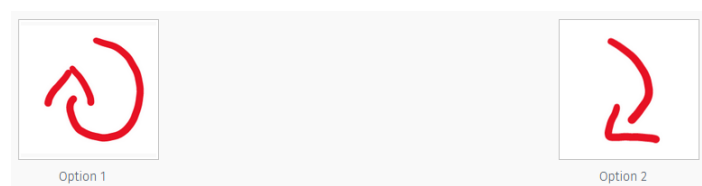
6. "Which annotation would you perform to zoom-in on an element?"



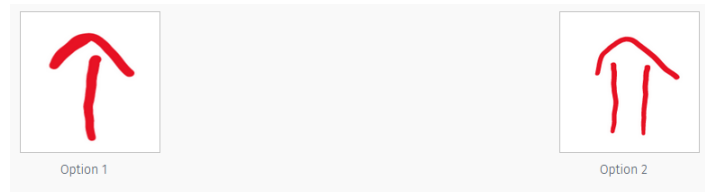
7. "Which annotation would you perform to rotate an element to the left?"



8. "Which annotation would you perform to rotate an element the right?"



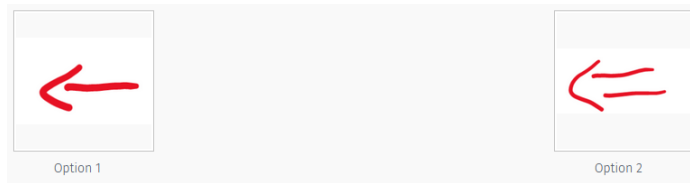
9. "Which annotation would you perform to move an element upward?"



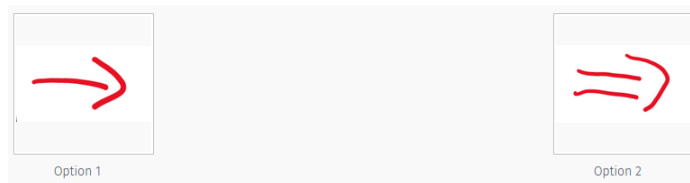
10. "Which annotation would you perform to move an element downward?"



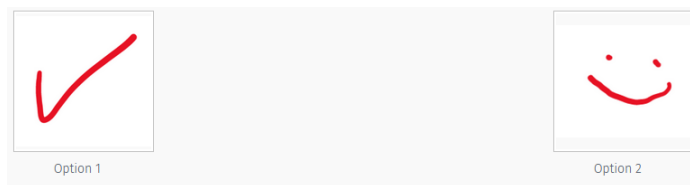
11. "Which annotation would you perform to move an element to the left?"



12. "Which annotation would you perform to move an element to the right?"



13. "Which annotation would you perform to annotate that this element is well designed?"



3.3.2 Participants' Information

To make sure our data is accurate, we asked participants for their level of usage of certain devices, their gender, and since we used QuestionPro ², a cloud-based survey and research platform that allows users to create and distribute online surveys, polls, and forms, we were able to get the country that the participants did the survey from. Tables 3.5 and

²<https://www.questionpro.com/>

table 3.6 show the demographic information of the 59 participants who participated in our survey. An example of this study survey can be found in section QuestionPro.

ID	Time(s)	Country Code	Age	Gender	Smartphone	Computer	Tablet
1	52	Belgium	23	Male	7	5	2
2	177	Lebanon	30	Male	6	5	3
3	106	Belgium	28	Female	5	5	2
4	132	Belgium	37	Female	6	5	2
5	106	Belgium	39	Male	6	5	2
6	148	Lebanon	23	Male	5	3	1
7	192	Lebanon	23	Male	6	5	3
8	79	France	22	Male	7	5	5
9	342	Lebanon	21	Male	2	2	4
10	106	Belgium	57	Male	6	4	3
11	124	France	23	Female	7	7	2
12	132	United States	33	Female	2	6	1
13	99	Poland	33	Male	7	7	1
14	80	Netherlands	21	Male	7	3	7
15	100	United States	43	Female	7	7	2
16	72	Netherlands	23	Male	6	7	1
17	125	United Kingdom	21	Male	6	7	1
18	67	United States	25	Female	7	7	2
19	329	Australia	21	Female	7	7	2
20	140	Singapore	24	Male	7	7	5
21	101	United States	32	Male	7	7	1
22	453	Greece	33	Male	7	4	1
23	145	United States	25	Male	7	7	1
24	87	United States	35	Female	7	7	2
25	69	United States	28	Male	7	7	2
26	84	United States	32	Female	7	7	3
27	63	United States	29	Male	7	5	1
28	100	Sweden	22	Male	7	7	1
29	97	United Kingdom	19	Female	7	7	1
30	136	United States	18	Male	7	7	3

Table 3.5: Participant Demographics in the Identification Survey: Sample Size of 59 Participants

ID	Time(s)	Country Code	Age	Gender	Smartphone	Computer	Tablet
31	73	United States	28	Female	5	5	1
32	91	Australia	30	Male	7	6	1
33	116	Luxembourg	18	Male	1	7	1
34	103	Brazil	22	Male	7	5	1
35	76	United States	33	Male	7	7	2
36	654	United States	49	Male	7	7	5
37	103	Australia	22	Female	7	7	1
38	93	Denmark	22	Female	7	7	7
39	76	United Kingdom	24	Female	7	4	1
40	174	United States	26	Female	7	7	2
41	293	United States	34	Female	7	7	5
42	123	Austria	19	Male	7	7	1
43	140	Austria	28	Male	7	6	1
44	76	Canada	30	Female	7	7	5
45	236	United Kingdom	25	Male	7	1	1
46	119	United Kingdom	33	Male	7	7	7
47	131	United States	46	Male	7	7	3
48	258	United States	33	Male	7	7	1
49	102	United States	44	Female	7	7	5
50	271	United States	19	Male	7	7	7
51	62	United States	36	Male	7	7	1
52	77	United States	21	Male	7	7	3
53	72	Canada	22	Male	7	7	1
54	159	Canada	21	Female	7	7	1
55	171	United States	22	Male	5	6	1
56	118	Sweden	23	Female	7	4	1
57	98	United Kingdom	22	Female	2	7	4
58	145	Sweden	18	Female	5	7	6
59	99	South Africa	45	Male	6	5	3

Table 3.6: Continued. Participant Demographics in the Identification Survey: Sample Size of 59 Participants

In our elicitation section, we discussed the importance of having a gender distribution as well as knowing the level of technology usage among the participants. In this part, we are going to use the same concept used in figure 3.4 and figure 3.5. Figure 3.11 shows the pyramid chart that represents the diversity of male and female participants who took part in our survey. Then, figure 5.5 shows a Likert chat of the agreement scaling of the participants involved. Since many users have switched to larger screen phones or lightweight laptops as their go-to device for personal and professional use, it has been reflected in our research findings, where we can see a relatively low number of participants who use tablet devices as shown in figure 3.5 and figure 5.5. Furthermore, having participants from different cultures ensures that the identification survey is not limited to a specific group of people, so it is a good approach to gather perspectives on how annotation types are interpreted in different parts of the world, this enhances the quality of our data.

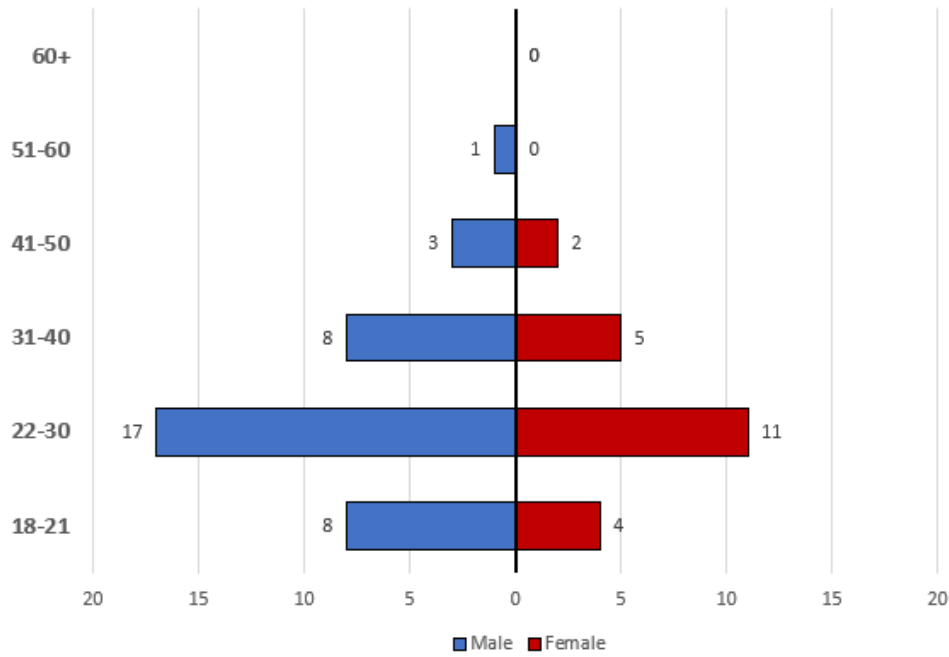


Figure 3.11: Age Distribution of Participants in the Survey of Identifying Annotation Types

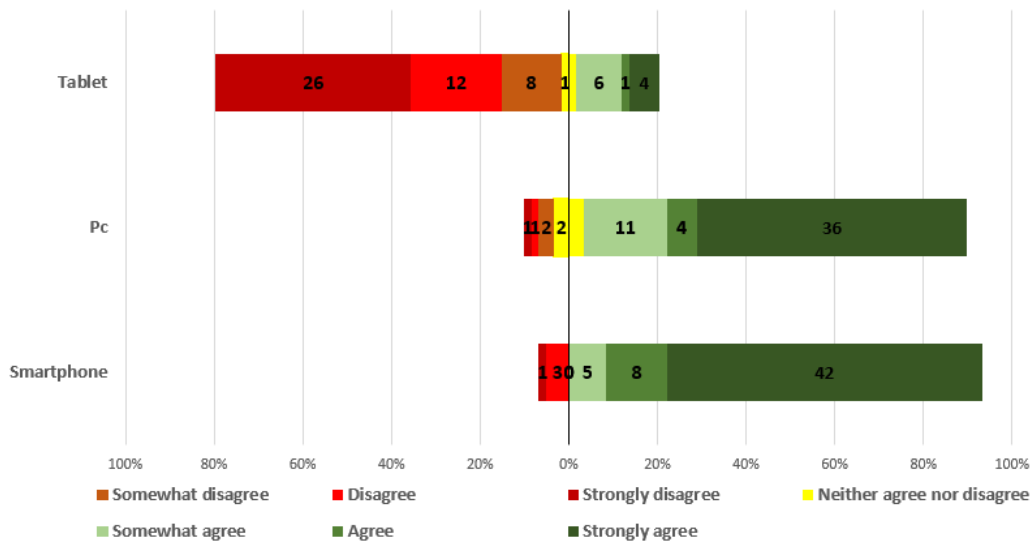


Figure 3.12: Likert Chart on Agreement Scaling - Identifying Annotation Types

3.3.3 Results

In this survey, we presented participants with different scenarios using images of annotation types as options, some questions had 2 options, others had 3 to choose from, and we asked them to select only the option that they considered the most suitable for a given task. This approach provided us with a good understanding of how participants perceived different annotation types, which, in turn, would help us design better user interfaces. The survey conducted took participants an average time of approximately **142** seconds. The results collected (see Table 3.7) showed that certain annotation symbols were selected

more frequently among participants, indicating a shared understanding. In addition to that, the annotations produced within a project are scattered by sets [15], and based on that we used the results to identify set of annotation types (see Figure 3.13) that were mostly chosen by the participants.

Question N°	Option 1	Option 2	Option 3
1	32.20%	67.80%	-
2	94.92%	5.08%	-
3	72.88%	27.12%	-
4	61.02%	38.98%	-
5	49.15%	6.78%	44.07%
6	38.98%	10.17%	38.98%
7	49.15%	50.85%	-
8	62.71%	37.29%	-
9	88.14%	11.86%	-
10	86.44%	13.56%	-
11	88.14%	11.86%	-
12	86.44%	13.56%	-
13	88.14%	11.86%	-

Table 3.7: Participant Responses: Percentages of Chosen Options per Question

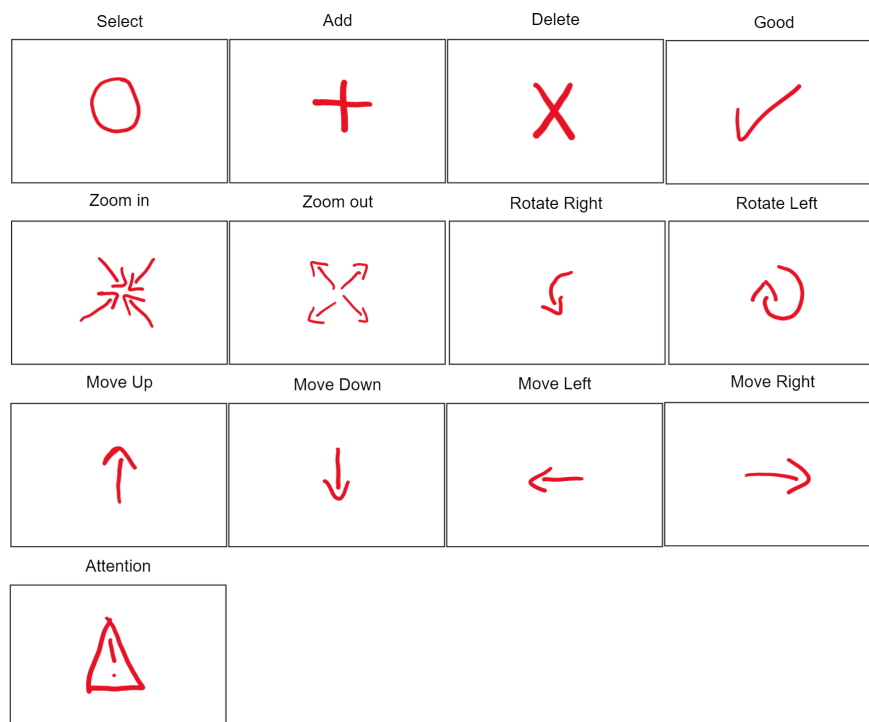


Figure 3.13: Identification Survey Results: An Overview of the Identified Annotation Types

3.4 Conclusion

In conclusion, this chapter discussed the work that was done to define an effective set of annotation types. Through crowd elicitation and an identification survey, we were able to gather information and identify various types of annotation that were analyzed on *AGATe* [49], which showed a high level of agreement between participants for certain annotations, while others showed lower rates of agreement, highlighting the importance of conducting such studies to ensure clarity. Through this analysis, we were able to identify patterns in how users interpreted certain symbols, which allowed us to refine our annotation types to be more useful in the future. In the next chapter, we will introduce "*Crowditivity*", a platform that we built to elicit requirements from the user interface, where users have the ability to create requirements, annotate images, and recognize types of annotations.

Chapter 4

Development of *Crowditivity*

This chapter will introduce *Crowditivity* ¹, a platform that we developed to elicit user interface requirements. In the first part, we will discuss the process behind the development of this platform, the requirements needed, and the stack that was used. Then, we will illustrate the architecture in the form of Unified Modeling Language "UML" diagrams to understand how the specification meet the requirements. Then, we will gather the information and use it to understand how the implementation of *Crowditivity* was done. Finally, we will introduce the $\$P+$ [47] algorithm that was used to recognize the annotations that were created on an image.

4.1 *Crowditivity*

In an elicitation study, a crowd of end-users is needed to perform some activities, and based on that, an analysis is done on the results. *Crowditivity* which means "Crowd Activity" was designed to help with the development process of a software and ensure an implementation of an interface that meets the needs of the users. The process behind the development of the platform will be divided into 3 main sections: **Conceptualization**, **Specification**, and **Implementation**. Figure 4.1 shows the landing page when a user visits the platform.

¹<https://crowditivity.web.app/>

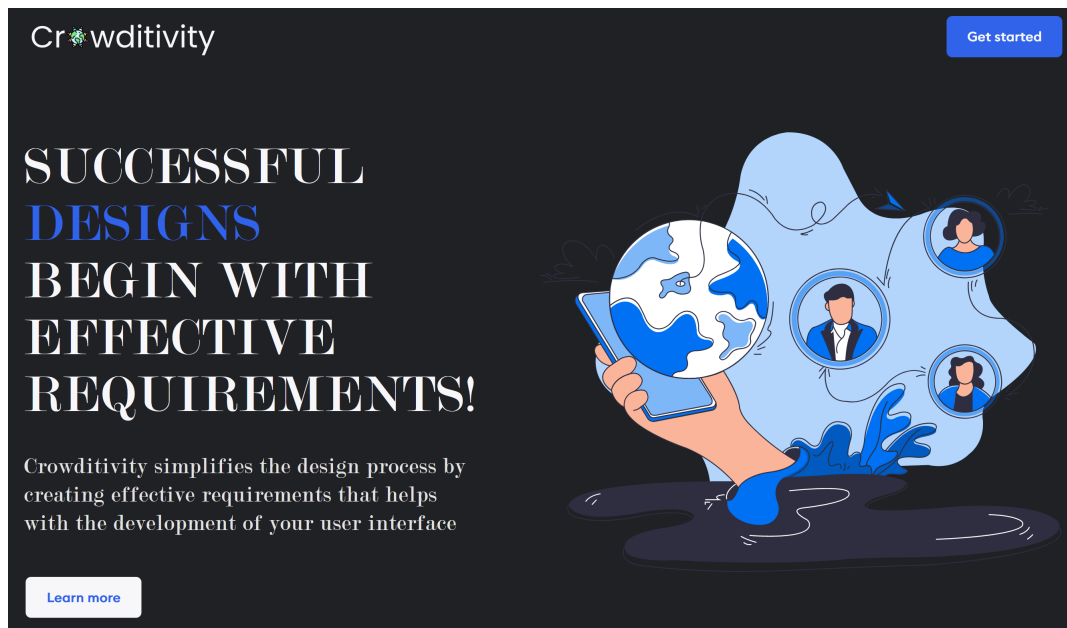


Figure 4.1: Landing Page - *Crowditivity*

4.2 Conceptualization

As mentioned above, *Crowditivity*'s main purpose is to gather user interface requirements from end-users. When developing a software, we can be lost in the process of how different parts should be connected with each other, what was requested in the beginning, and how the software should be developed to meet the requirements. Building a software involves people working on different sections to deliver a working product, but how can we make sure that what we are building is really what is intended to be? This is why documentation becomes a critical phase in the development life cycle. Working in a team to build a product requires all the stakeholders involved in the process to understand what needs to be done in each phase. In section **Software Development Life Cycle (SDLC)**, we introduced the process needed to develop a software defined as *SDLC*, with the first step being the "Requirement Analysis". The main objective of developing a platform like Crowditivity is to be used as a tool for identifying the requirements needed before going into the design and coding phases. Understanding what a software will look like and how different elements behave when connected with each other is important because if each individual working on the development would have their own ideas, it is not possible to have a structured design that can be relied on, and the software would look differently than how it was intended to look like in the first place. The interface is the first point of contact between the user and the software, and having a poorly designed one can lead to user frustration and poor user experience. To ensure a successful user interface, it is essential to have a good understanding of the user needs and preferences, so this is where good documentation of user interface requirements comes into place, by documenting in a structured way, developers can better understand the interfaces that meet the needs of the designers.

4.2.1 The Process

Crowditivity is a platform where end users, designers, and developers can use it to create, annotate, and review user interface requirements. To develop this platform, we need to understand the main functionalities that were implemented and what each one does. In *Crowditivity*, a user is able to create a study, annotate images that would help designers better understand their interfaces, and review their work once each of the aforementioned steps has been completed. To be able to use the platform, we will discuss in detail what each functionality is, the purpose behind it, and how it works.

4.2.2 Create Study

Elicitation studies can be seen as a middle ware between what was designed on a wireframe and what is actually coded. To prevent miscommunications between the designers and the developers working on the user interface, and to avoid misleading information about what will be done on this software, the "create study" functionality in *Crowditivity* is a way to prevent such problems from occurring. When users want to create a study, they are prompted with an email verification to be eligible to start, and once verified, the user can add a title, provide a small description of the study being created, and an image that would be shown as a reference preview of the software. After creating the study, the user is taken to a page where the main process of gathering user interface requirements begins. Before starting the process, as a first step, a short description of how to create effective requirements is shown. Then, the user is requested to add their demographic information, such as their: First Name, Last Name, Age, Gender, Education, and their nationality. In the third step, images can be added by the user where each one has its unique set of requirements that will be written in the fourth and last step. The last step involves creating requirements using the template "*VOLERE*" that was covered in section *VOLERE*. A *VOLERE* template has requirements that are specifically created to help designers in the development process. On the basis of how many images were chosen, a template is generated for each. Users have the option to add up to 5 supporting images to the main image, and these images are added to provide extra information related to a series of tasks or the relation that they have in common. Up to 15 unique *VOLERE* templates can be added to identify the requirements of an image. The fields in the *VOLERE* template are as follows:

- **Requirement Title:** a title used to identify different requirements
- **Requirement ID:** a unique Identifier for each requirement
- **Requirement Type:** can be functional requirement or non-functional requirement
- **Event number:** list of events
- **Use case number:** use cases that need this specific requirement
- **Description:** a one sentence statement of the intention of the requirement
- **Rationale:** A justification of the requirement
- **Originator:** Who raised this requirement

- **Fit criterion:** A measurement of the requirement such that it is possible to test if the solution matches the original requirement
- **Customer satisfaction:** Degree of stakeholder's happiness (from 1 to 5) if this requirement is successfully implemented
- **Customer dissatisfaction:** Degree of stakeholder's unhappiness (from 1 to 5) if this requirement is not part of the final product
- **Priority:** The relative importance of the requirement (Not important, Important, Very important, and Crucial)
- **Conflicts:** If another requirement might affect this requirement
- **Supporting Materials:** Pointer to document that illustrates and explains this requirement
- **History:** Creation date and if any changes were made on this requirement

4.2.3 Annotations

We use annotation to change elements in the design of an interface. Color-changing animation is utilized by some studies for feedback [5]. For that reason, we implemented in *Crowditivity* a way for users annotate their images by choosing one that has some modifications to be made on it, and then, the user is able to annotate using 3 different colors (Red, Green, and Blue) based on how important each change is, choosing the color "red" indicates that that this change has the highest priority among all the other annotations. The "green" and "blue" colors indicate that this change has a normal and low priority, respectively.

Having the ability to annotate is important in software design because it allows designers to highlight specific features that are relevant to the interface itself. In software development, it is often necessary to convey technical information to team members who may not be familiar with the topic of designing an interface, so annotating can be a solution to this problem.

The process of adding annotations on an image is described as follows:

- **Input mechanism:** is how the input from the user is being collected, this process involves the use of a mouse or trackpad for effective annotations.
- **Document type:** the format of the document to be annotated on should be image of type (.png, .jpeg, .jpg, .svg).
- **Overview of adding annotations:** is the process of collecting multiple user-generated inputs on an image to be used in the recognition part.
- **Annotation types recognized:** based on a pre-defined template that we created, the annotations created by users are recognized and added to a list of recognized inputs.

In our platform, annotating an image has 3 main steps:

- **Annotation name:** the user chooses a name for the image that will be annotated

- **Marking up:** adding annotation symbols that were discussed in section [Elicitation Study of Annotation Types](#) using 3 different colors (red, green, or blue) to an image of their choice
- **Data transfer:** the user uploads the image with its annotations to the database which will be accessible on the user's "My Annotations" page.

When designing software, it is often necessary to convey complex information in an understandable way, and this can be particularly challenging when dealing with complicated user interfaces, which can be difficult to understand without any additional information. In general, designers use annotations to create clear, concise, and effective visual representations of complex information.

4.2.4 Review the work

We talked about the functionalities of *Crowditivity* where users can create their own studies and annotate different kinds of images to have a better understanding of the software that requires some changes in its design. In *Crowditivity*, each user has their own account for creating studies and annotations, this helps the user to keep track of their work. By allowing users to see what they have done, they can better understand their progress, which then leads to good user engagement, as they feel more invested in their experience and are more likely to continue using the platform. *Crowditivity* lets its users to:

- **Check studies:** ability to show the user the studies that they created
- **Check annotations:** ability to show the user all their annotated images
- **Check requirements:** ability to show the user all the requirements and their respective images that were created in each study

4.3 Specification

When developing an application, the specification phase plays an important role, as it follows the conceptualization phase. In this phase, we outline the application's scope by creating detailed class and activity diagrams. The outcome of this phase results in a detailed functional specification document that projects the technical requirements. The specification provides a clear and concise plan to guide the development process throughout the application implementation phase.

4.3.1 Class Diagram

Class diagrams offer a visual representation of everything related to the classes, attributes, and methods that make up the application, giving a perspective on its architecture. By creating a graphical representation of the classes, it becomes easier to perceive the overall functionalities of an application. In addition to that, class diagrams serve as a tool for showcasing information about the application to different stakeholders, including developers, designers, and managers. They identify potential areas of improvement within the application itself. By analyzing the relationships between classes, we are able to identify areas that are complex, which can be utilized to enhance the overall quality and make the

application more maintainable. These types of diagram assist in creating better quality software that are more efficient and understandable. We will introduce the "Conceptual" and the "Logical" diagrams that were created to better understand the architecture of *Crowditivity*:

- **Conceptual diagram:** identifies the main entities and relationships involved in the database, without getting into the details of their implementation.

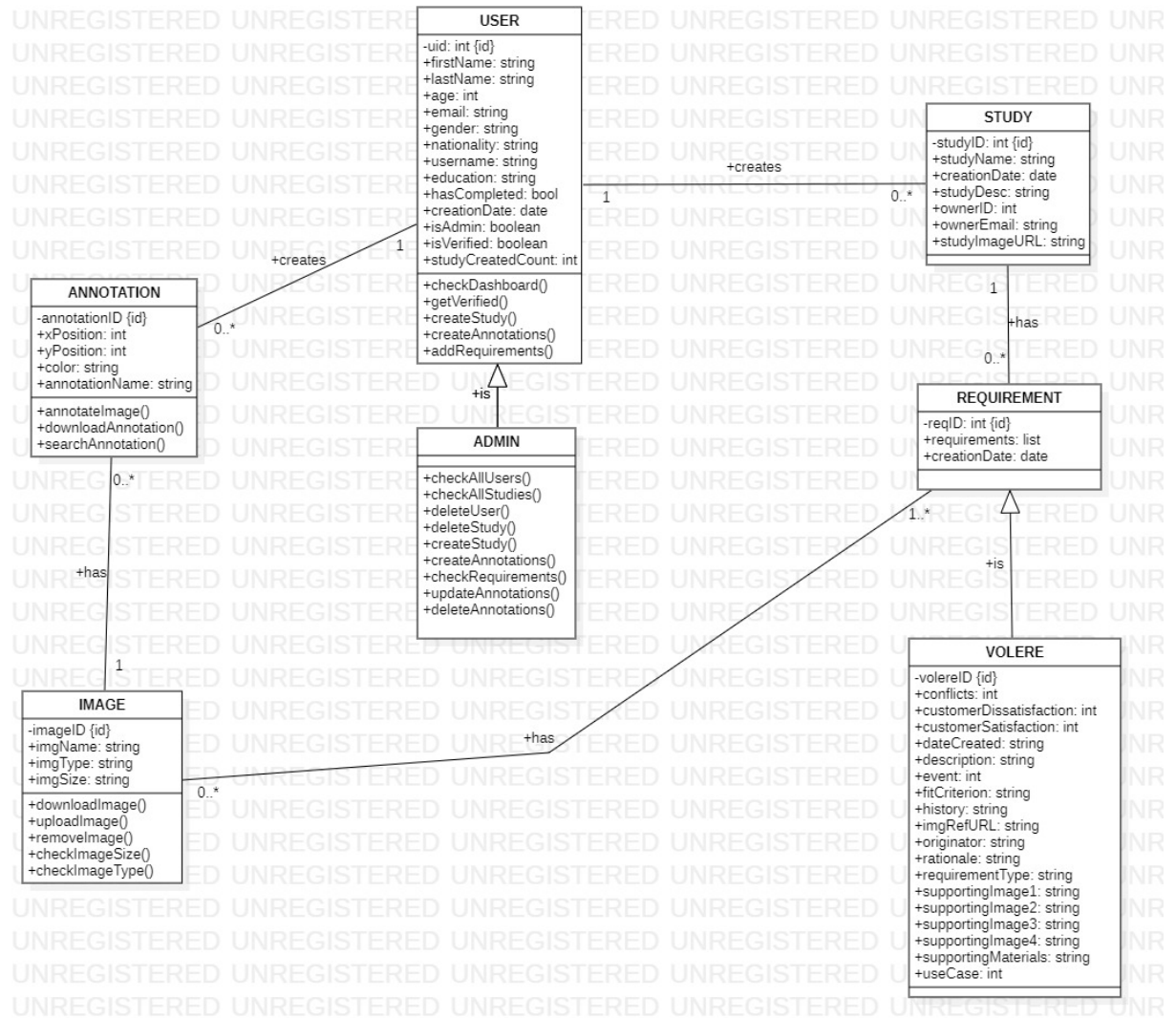


Figure 4.2: Conceptual Diagram - *Crowditivity*

- **Logical diagram:** illustrates the logic of the relationships between the entities, attributes, and tables that related to each other.

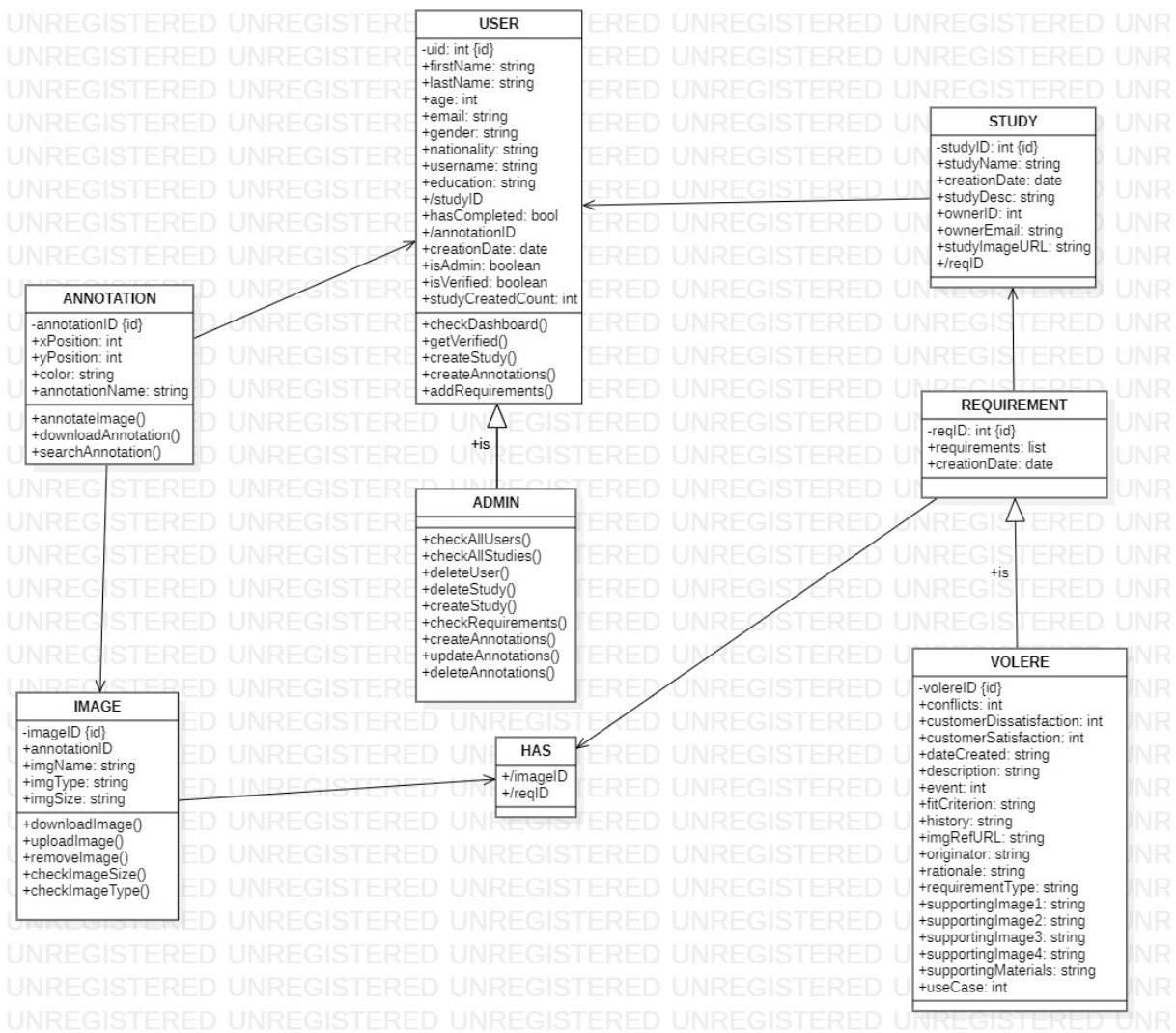


Figure 4.3: Logical Diagram - *Crowditivty*

4.3.2 Activity Diagrams

Activity diagrams are tools for visualizing the flow of actions in software, allowing stakeholders to see how various components interact with each other. A key benefit for using these activity diagrams is their ability to break down complex processes into smaller, more manageable ones, which then helps in understanding how each component contributes to the overall software. By creating multiple activity diagrams and having each one focused on specific parts of the application, it helps in the visualization of how the system behaves. Moreover, having multiple activity diagrams create a common way for discussing the software's complexities, which can result in a better understanding of the overall development process.

Main Activity Diagram

Activity diagram 4.4 gives a general overview of the tasks that normal and administrator users are able to perform in *Crowditivty*. In this case, the user can either be and admin or a normal end-user. The users start by authenticating themselves on our platform, if

errors occurred along the way, the user can try to authenticate again. When a successful authentication happens, users can check their dashboard, in this case being a normal user or an admin user, both users have the ability to check their dashboard. In the dashboard section, users are able to create a new study, add requirements to an existing study, or create new annotations. Now, if the user is an admin, they can perform the tasks of a normal user, and in addition to that, they are able to manage all the users and studies on the platform. Finally, when users are done with their work on the platform, they have the ability to log out of their session.

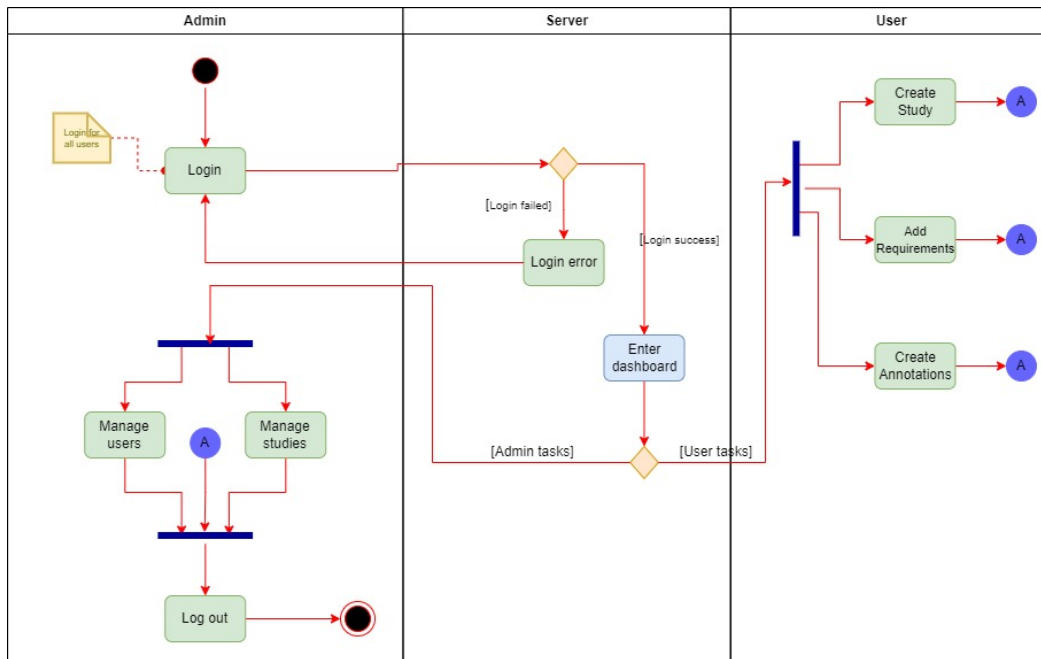


Figure 4.4: Main Activity Diagram - *Crowditivity*

Create Study - Activity Diagram

In this activity diagram 4.5, the user can be either an administrator or a normal user. Users starts by verifying their email addresses based on an email they received to their inbox when they first joined the platform, if any error occurs in this step, the user can try to verify their email address again. When users are verified, they can check their study dashboard. In this dashboard, users are able to create a new study by adding the study details such as the "Title" and "Description", after which they are able to add an image that refers to their study objectives. When all the details of the study are completed, users can upload the study to the database and if no errors are returned from our server, the user can add requirements to their newly created study. Finally, when users are done with their work on the platform, they have the ability to log out of their session.

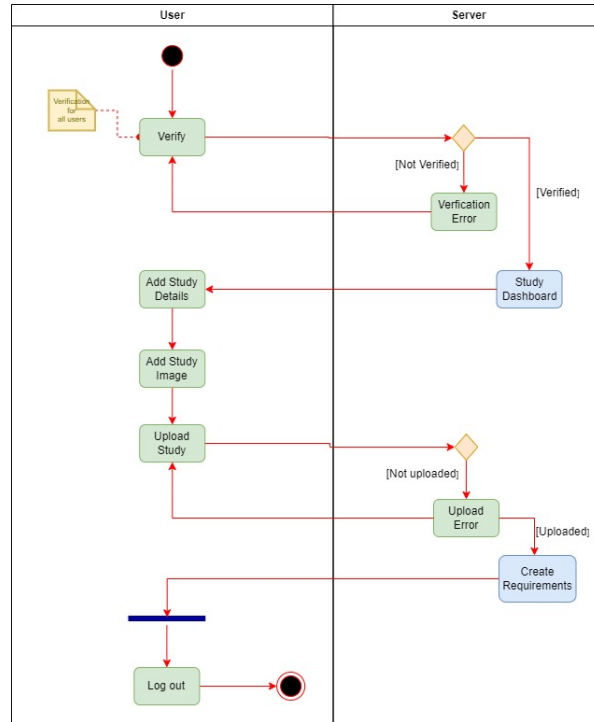


Figure 4.5: Study Activity Diagram - *Crowditivity*

Create Requirement - Activity Diagram

In this activity diagram 4.6, the user can be either an administrator or a normal user. After the user created their study, they are able to create new requirements, by understanding the process required to create requirements, after that they can add their personal information, then, they are able to add images to create requirements from, and if the image size is respected, the user can now start filling the *VOLERE* fields to create requirements as well as being able to add supporting images to their main image. When all the details of the requirements are completed, users can upload them to the database, and if no errors were returned from the server, the user can check their requirements dashboard to see all their requirements. Finally, when users are done with their work on the platform, they have the ability to log out of their session.

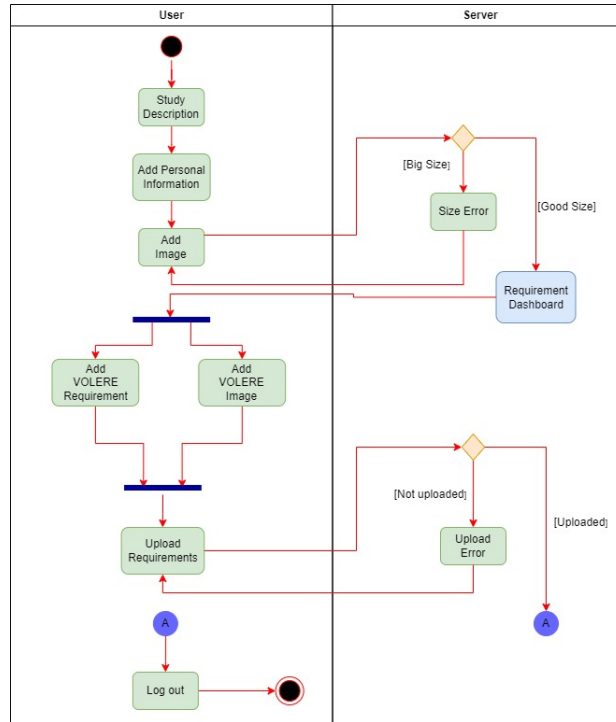


Figure 4.6: Requirements Activity Diagram - *Crowditivity*

Add Annotation - Activity Diagram

In this activity diagram 4.7, the user can be either an administrator or a normal user. Users are able to choose an image to annotate on; then if the image size is respected, they can start creating annotations. They have the ability to choose the color for their annotations, annotate the image, and add a name for their image. After completion, users can upload their annotated image to the database, and if no errors returned from the server, they can check their dashboard to see all their created annotations. Finally, when users are done with their work on the platform, they have the ability to log out of their session.

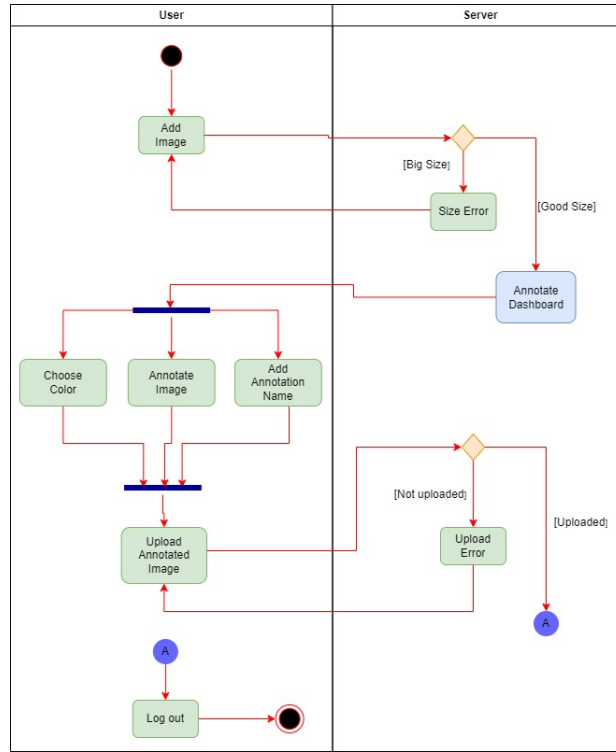


Figure 4.7: Annotations Activity Diagram - *Crowditivity*

Admin Tasks - Activity Diagram

Activity diagram 4.8 is related to the tasks that an administrator can perform on the platform. After the user has successfully verified themselves as admin users, they are able to see their admin dashboard. In this dashboard, you can verify a user or delete a user. Admins are also able to delete annotations and requirements created by users. These tasks are requested from a user if for some reason they had some issues verifying themselves or deleting their work on the platform. Once the admin chooses which task they want to perform, and if no errors were returned from the server, the user is either verified or deleted, and either an annotation or a requirement is deleted from the database. Finally, when admins are done with their work on the platform, they have the ability to log out of their session.

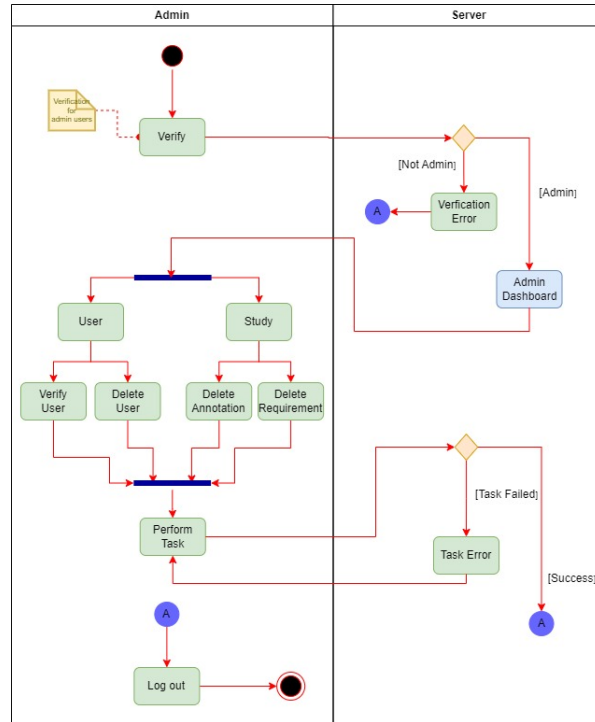


Figure 4.8: Admin Activity Diagram - *Crowditivity*

4.4 Implementation

This section will be focused on how *Crowditivity* was developed. We will define the technology stack that was used to develop it and give a walkthrough of the different functionalities that were mentioned in the [The Process](#) section. During the implementation phase, it is important to balance functionality with usability, so we must ensure that the system is capable of doing all the required tasks and also making sure that the interface is user-friendly.

4.4.1 Technology Stack

Choosing the right technology stack can be a problem for any software development projects because it involves an extensive research about all the possible technologies that are present. The technology stack consists of the technologies that were used to build and deploy our platform, such as programming languages, frameworks, and database. By choosing the right stack, we can ensure that we are building a high-quality software that meets the needs of its users. Figure 4.9 shows a well-defined plan for which technology is needed and how each functionality should behave. When building a website from scratch, the technology stack chosen is of high importance as it determines the website's performance, scalability, and user experience.

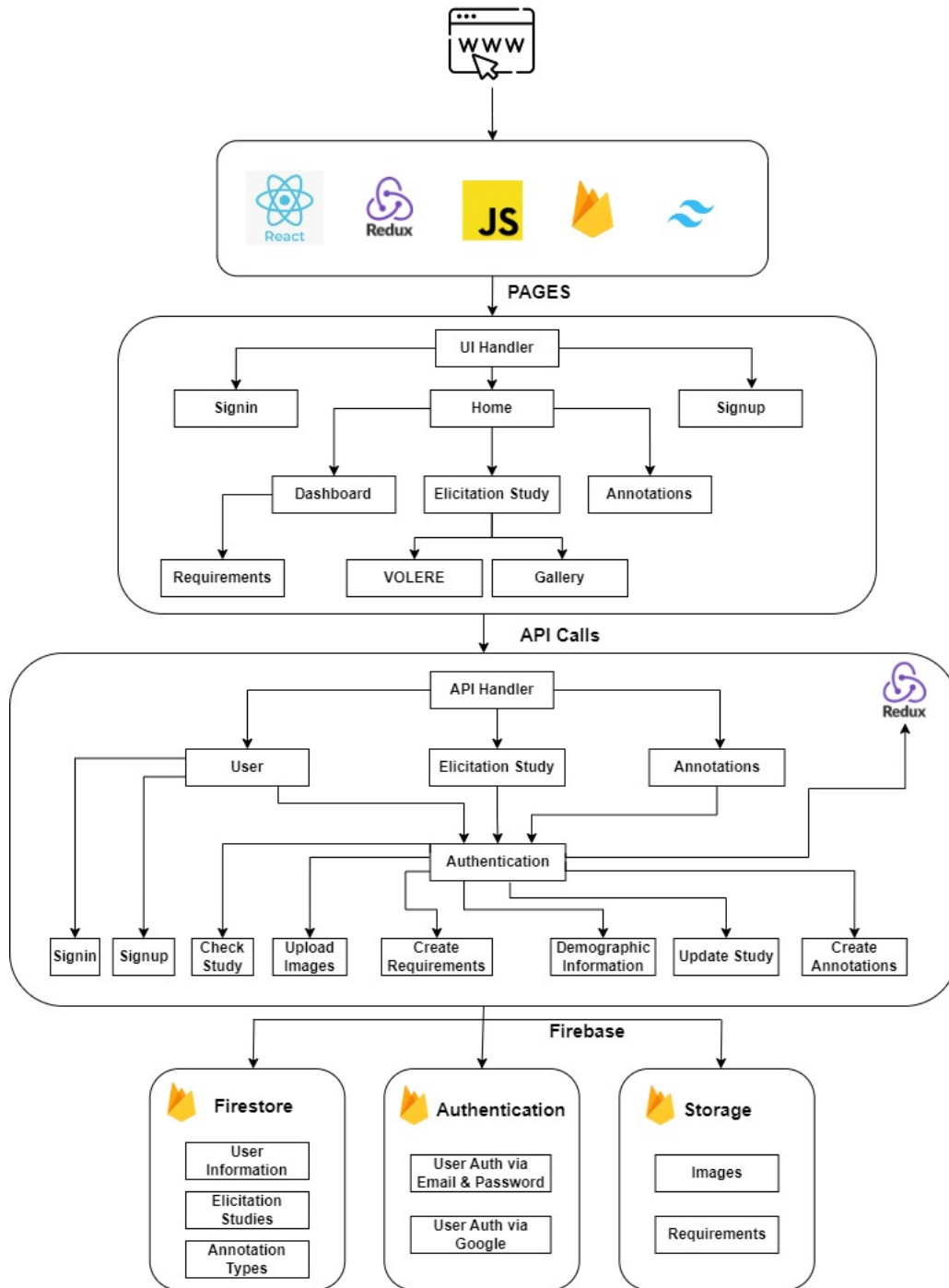


Figure 4.9: Technology Stack Diagram - *Crowditivity*

Front-end

For the front-end part of *Crowditivity*, we decided to work with these frameworks:

- **ReactJS:** a popular and widely-used JavaScript library that helps in building user interfaces in a professional way. ReactJS has a lot of good practices, it is easy to use, and it supports reusable components, which help in writing the code once and reuse it multiple times in different places in the application, which can significantly reduce development time. It provides good performance, this is due to its implementation of Virtual DOM, whose primary objective is to enhance the overall performance of

the application [20], unlike other traditional HTML which is built on a real DOM. It updates only the changed components, which means reducing the number of renders and improving the website's loading speed. However, as any other framework, ReactJS has its own downside, for instance, ReactJS uses JSX, which is JavaScript + XML, that allows us to write HTML-like syntax in the JavaScript code. JSX can be seen as a language that is confusing to work with, specially in large projects. In addition to that, when the application grows, additional tools are required, that being said, extra libraries are needed to build a complete the desired functionalities, this can add complexity to the development process.

- **TailwindCSS**: a utility-first CSS framework that has pre-defined user interfaces classes to be used on the fly. These pre-built classes can be used to create designs without having to write all the CSS from scratch. Tailwind CSS also provides a high level of customization, allowing us to customize the default styles of the ReactJS application. Tailwind CSS is a utility-first approach. This approach means that each CSS class is specifically built to perform specific results, making it easier for us to style the user interface components. Having to think about how to create a responsive application can be time-consuming and a lot of additional work in the development process, but Tailwind CSS includes responsive design classes, which allows us to create responsive user interfaces that can adapt to different screen sizes. However, we need to take into consideration that the classes that Tailwind CSS provides are well designed in a way that is able to change them to fit your own needs. This limits the customization options, so a good understanding of how CSS attributes work to develop custom classes is beneficial.

Back-end

For the back-end part of *Crowditivity*, we decided to work with these technologies:

- **Firebase**: a cloud-based platform for web development that has many useful features like cloud storage, authentication, and hosting. We decided to choose Firebase because it provides a scalable infrastructure that allows us to focus on creating the website's functionalities rather than spending time on managing the database infrastructure from scratch. Firebase provides a high level of security as well as authentication features that can help protect users' data from any DDoS attacks or hackers trying to break into the application to get users' personal information. Hosting on Firebase is really helpful when we want the platform to be accessible by users when we launch it in production phase. Firebase has automatic backups and recovery features, which can help protect against data loss in case of system failure. Since the main focus is developing a platform to gather user interface requirements, the use of Firebase functions, such as "Analytics", helps us understand user behaviors, engagement, and how they are using the application. This can be used to improve the user experience. In addition to that, one of the disadvantages of Firebase is that it may not be suitable for complex data querying, which makes it difficult to structure the data in a way that can be optimized and result in slower query times.
- **Redux**: is an easy-to-use predictable state container [10]. It is a state management library that works along with ReactJS to help manage the application's state in an efficient way. The main concept of ReactJS is having stateful components that are rendered when needed, so it is like a centralized store where the state of the

entire application is stored. Using Redux can be beneficial because of its ability to manage states in an organized way. When the application fetches data from Firebase, it typically makes an asynchronous API call to retrieve these data, so this API call takes time and if the same data is requested multiple times, it can result in unnecessary API calls. Redux is a great solution for these problems, by storing the fetched data in a global state container that is accessible any time and in any component of the application. This can improve the performance of the application. This being said, a drawback of using a state management library such as Redux is that overusing it to store every state can lead to performance issues. Too much data stored in the store can slow the application and increase memory usage.

4.4.2 *Crowditivity* Walkthrough

To better understand how *Crowditivity* works, in this section we are going to do a walkthrough on the different aspects of *Crowditivity*.

Authentication

We start first by authenticating the user, as we can see in figure 4.10 and 4.11, the user must create an account (if they haven't created one already) by authenticating via their own email and password, or by using Google authentication's service where they are automatically verified as users who are eligible to use the functionalities of the website.

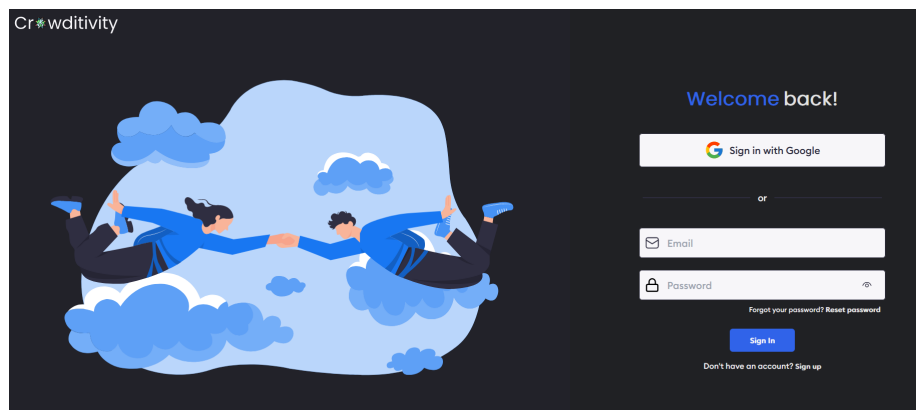


Figure 4.10: Sign in Page - *Crowditivity*

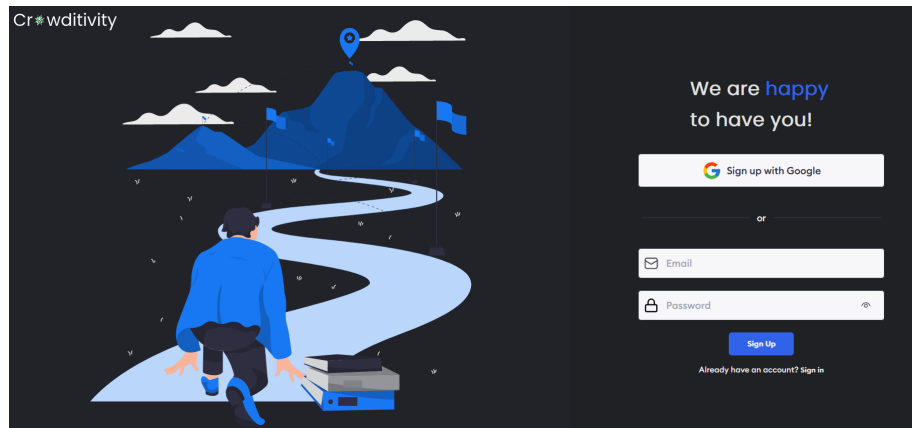


Figure 4.11: Sign up Page - *Crowditivty*

Create Study

Once the user is authenticated, the "Create Study" page shown in figure 4.12 lets the user start the process of creating a study to gather requirements. Earlier, we mentioned the purpose of creating a study that allows users to generate user interface requirements using the template *VOLERE*.

Figure 4.12: Create Study - *Crowditivty*

Create Requirements

After the study has been created, we generate a unique identifier for this study that can be accessible by any user at any time. The process for creating a successful study can be done based on a series of steps that will be explained in detail:

- **Step 1 - Description:** Figure 4.13, the user is first prompted with a descriptive explanation of how to create requirements in an correct way in the upcoming steps

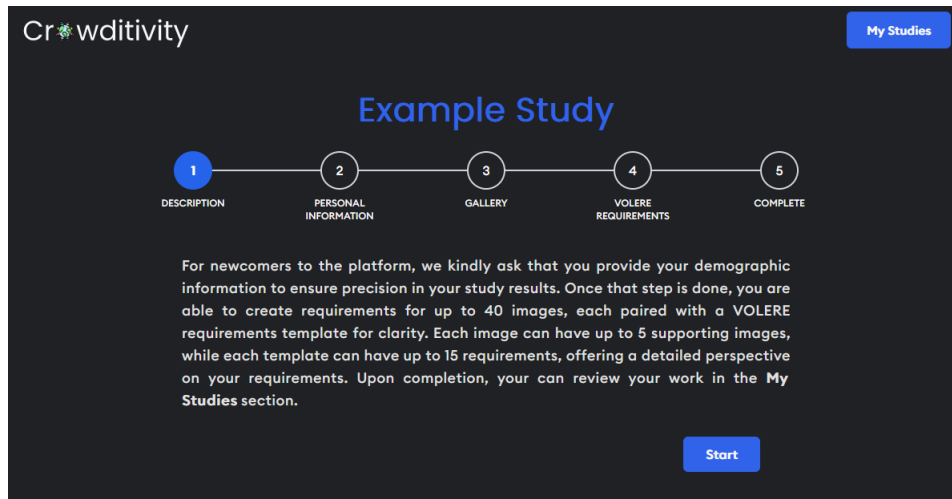


Figure 4.13: Study Description - *Crowditivty*

- **Step 2 - Personal Information:** Figure 4.14, the user is then requested to fill in their demographic information that are used to identify differences in user preferences across different age groups, genders, or educational level. This helps to have a better understanding of the users who perform the study to develop more effective strategies to meet their needs.

Figure 4.14: Personal Information - *Crowditivty*

- **Step 3 - Gallery:** Figure 4.15, this step requires the user to choose images that they want to create specific requirements for. Each image must respect the required size, which is 5 MB, and the image type (PNG, SVG, JPEG, JPG).

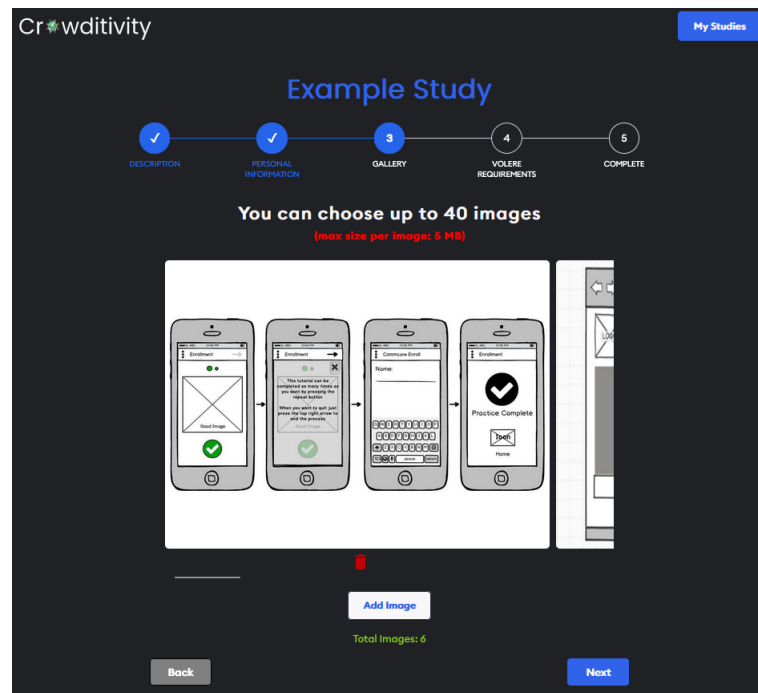


Figure 4.15: Choosing Images - *Crowditivty*

- **Step 4 - VOLERE:** Figure 4.16, based on the images that were chosen in step 3, we can start by creating up to 15 requirements using the template *VOLERE* discussed in section *Create Study*. In the example below, we chose an image and provided five requirements that we think would help tailor a good user interface for a system.

Figure 4.16: *VOLERE* Template - *Crowditivty*

- **Step 5 - Complete:** Figure 4.17, the last step in creating requirements is when

the user clicks the Submit button to store all the images and requirements that were created in the database.

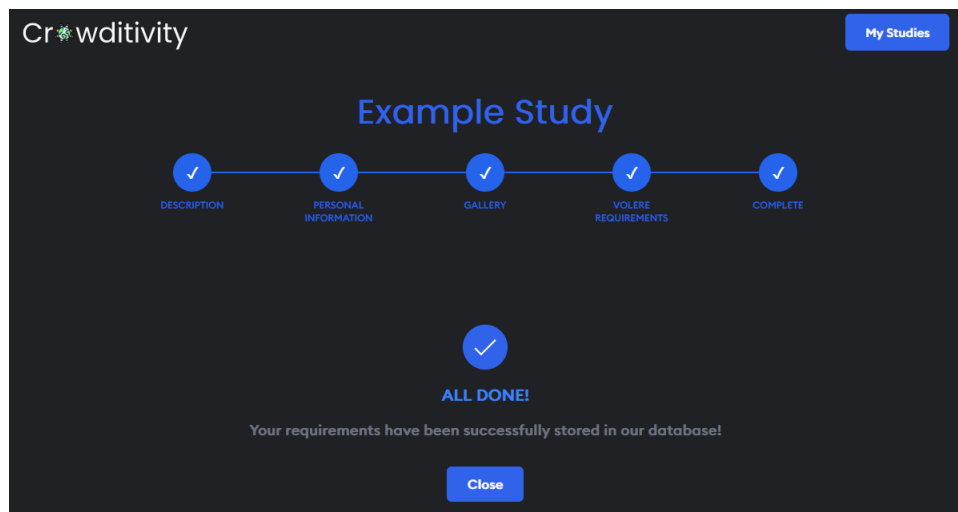


Figure 4.17: Complete - *Crowditivty*

Annotate Images

This section will be focused on understanding the process of annotating an image. First, the user is welcomed with a page that prompts them to choose an image that they want to annotate on see (as in Figure 4.18).

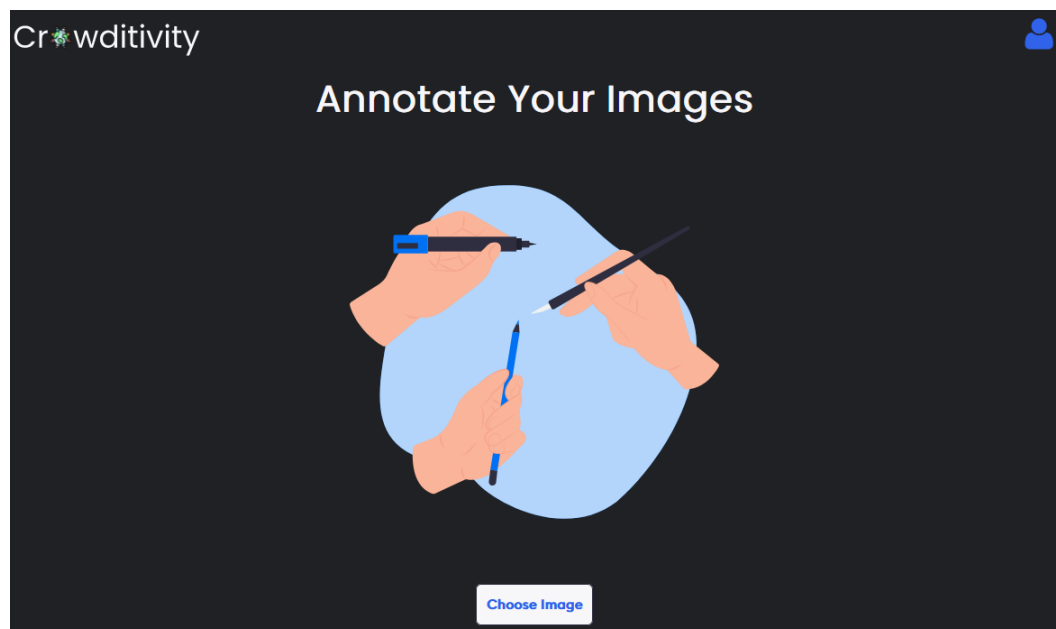


Figure 4.18: Add an Image - *Crowditivty*

When a user chooses an image, they are able to choose between 3 different colors to annotate their image: By default the color Blue is selected, the color Green is used for important annotations, and finally the red is used for critical annotations to show that this change is very important. Figure 4.19 shows an example, where the logo "Boogle" is selected. In addition to that, we can clearly see that the annotation that was used is a

circle (of type "select") which is a type that was pre-defined in Chapter 3. We integrated in *Crowditivity* a way to recognize the inputs created by the user to better understand which annotations represent which predefined type. In section [Classification of Data](#) we discussed the different annotation types in detail.

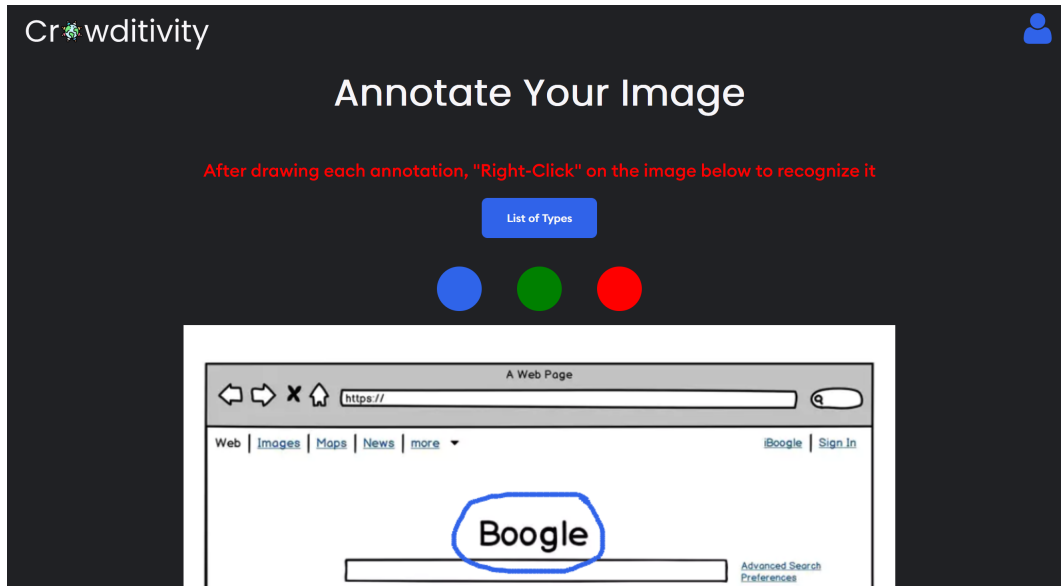


Figure 4.19: Annotate Image - *Crowditivity*

In the upcoming section, we are going to introduce $\$P+$, a 2-D gesture recognizer that is used to recognize user-generated annotations on an image.

4.4.3 The $\$P+$ Recognizer

$\$P+$, is an updated algorithmic design for the $\$P$ recognizer, that improves recognition accuracy and execution speed for gestures produced by people with low vision [47]. Figure 4.21 shows the differences in drawing a five-point star between people without visual impairment compared to people with this condition. To have a maximum accuracy when recognizing gestures an updated version of the $\$P$ algorithm was introduced called $\$P+$. The $\$P$ recognizer was introduced by Vatavu et al. [48] as an articulation-independent gesture recognizer [47]. The name is derived from the *$\$$ -family* of recognizer algorithms, and the "P" means that this algorithm is a point-based recognizer so it makes use of "Point clouds". $\$P+$ introduced 2 main upgrades from its previous version: optimal alignment and exploiting the shape of the point cloud. The optimal alignment between two points works by having each point from a cloud to be matched with another point that has not been matched before because they are represented as the optimal option when points are matching. However, the upgrade that was made in $\$P+$ was, instead of matching the first point from a cloud to a "unmatched" point from another cloud, the algorithm would now match the closest points even though they were already matched earlier, this change resulted to be faster since the time complexity has decreased by approximately **34.1%** from $O(n^{2.5})$ to $O(n^2)$. The second change was to exploit the shape structure, since $\$P$ was creating a set of 2D gesture points without any ordering, which means that the algorithm ignored any connections between point a and point b. In $\$P+$, they introduced a solution to this problem by adding angles for matching points to improve recognition

accuracy. Turning angles capture the curvature of a gesture, which is an important feature for accurate recognition.

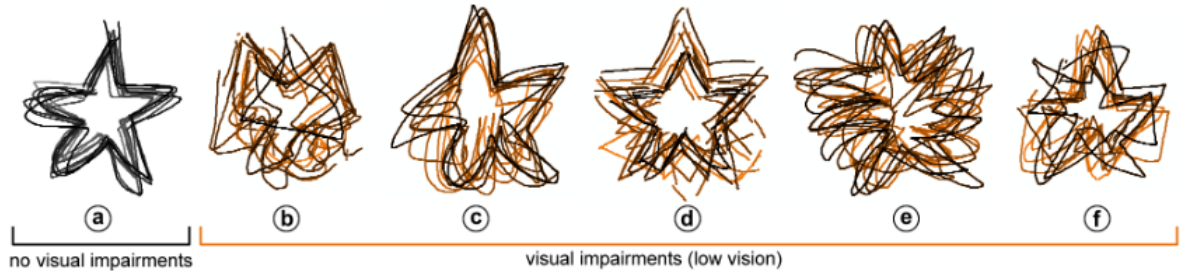


Figure 4.20: Gesture articulations produced by people with low vision present more variation than gestures produced by people without visual impairments, which negatively affects recognizers' accuracy rates. Ten (10) superimposed executions of a "star" gesture are shown for six people: a person without visual impairments (a); three people with congenital nystagmus and high myopia (b), (c), (d); and two people with chorioretinal degeneration (e), (f). [47]

Below we can see the pseudo-code used for the $\$P+$ recognizer [47]. This recognizer will recognize gestures that were drawn by comparing them to a predefined template. The normalize function is used to transform an input gesture to a standard size and position; this process makes it easier to compare it with other gestures. For each template gesture, it normalizes it to the same fixed length as the input gesture and then calculates the distance between these two gestures. The minimum distance is stored in a variable "d", and if this value is less than the current score, the score is updated to "d". Finally, the output of this algorithm returns the result and the score.

```

1 # $P+RECOGNIZER (POINTS C, TEMPLATES templates)
2 n <- 32
3 NORMALIZE(C, n)
4 score <- infinity
5 for all T in templates do
6 NORMALIZE(T, n) // should be pre-processed
7 d <- min (CLOUD-DISTANCE(C, T), CLOUD-DISTANCE(T,C))
8 if score > d then
9 score <- d
10 result <- T
11 return <result, score>

```

The pseudocode shown below is related to the cloud distance algorithm which is used in $\$P+$ to compute the distance between two point clouds. This algorithm allows one-to-many matches, so that each point in one cloud can be matched with multiple points in another cloud. The algorithm works by having for each point in cloud C find the closest match in cloud T and stores the index of the match in an array. After that, the distance between the matched points is added to the variable called "sum". Next, for points in cloud T that were not matched, it finds the closest match in cloud C and adds the distance to the same variable "sum" which is then returned as output.

```

1 # CLOUD-DISTANCE (POINTS C, POINTS T, int n)
2 matched <- new bool[n]
3 sum <- 0

```

```

4 // match points from cloud C with points from T ; one-to-many matchings
  allowed
5 for i = 1 to n do
6 min <- infinity
7 for j = 1 to n do
8 d <- POINT-DISTANCE(Ci , Tj )
9 if d < min then
10 min <- d
11 index <- j
12 matched[index] <- true
13 sum <- sum + min
14 // match remaining points T with points from C; one-to-many matchings
  allowed
15 for all j such that not matched[j] do
16 min <- infinity
17 for i = 1 to n do
18 d <- POINT-DISTANCE(Ci , Tj )
19 if d < min then min <- d
20 sum <- sum + min
21 return su

```

In Chapter 3 (see Section [Elicitation Study of Annotation Types](#)), we described in detail how we collected the annotation types from a group of participants through a crowd-elicitation process. We then identified different types of annotations based on the results that were elicited. After that, we needed to use these annotation types to train our models shown in Figure 4.21 for more accurate results for the $\$P+$ recognizer used in our platform. By doing so, we were able to create a pre-defined template of the different types that were used to provide better results for our users.

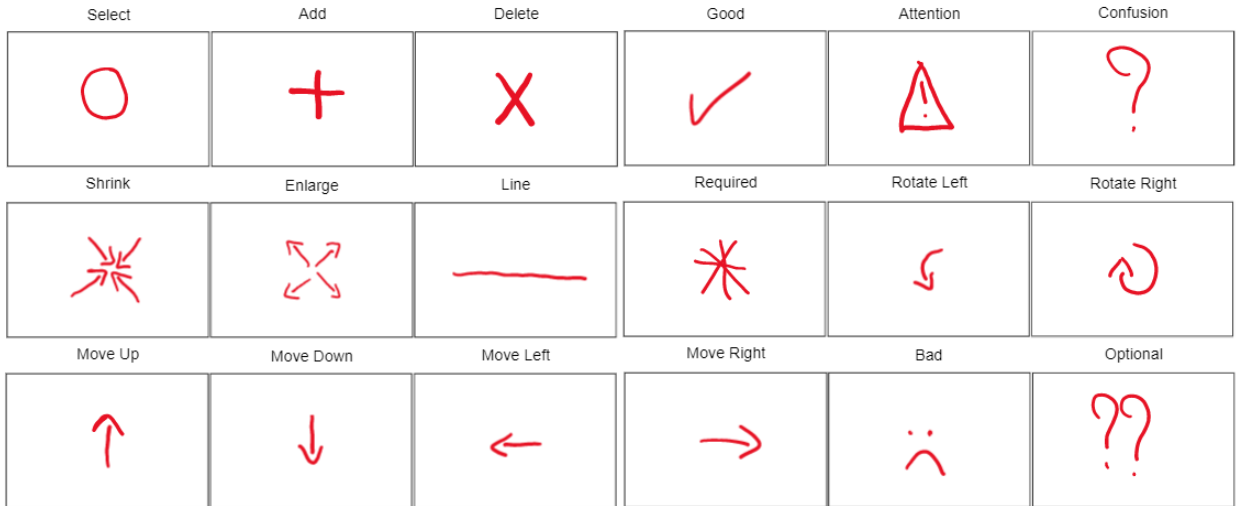


Figure 4.21: Annotation Types used for Recognition in *Crowditivity*

In Section [Annotation Types Recognition on Crowditivity](#), we present the recognition results of the 18 annotation types, effectively demonstrating the efficacy of the recognition feature and the use of the $\$P+$ algorithm that was implemented in *Crowditivity*.

4.5 Conclusion

The analysis presented in this chapter, gave a general overview of the underlying concepts behind *Crowditivity* and how it was developed. Three phases were described, the first one was "conceptualization", where the behavior, scope, and objectives of the platform were introduced. Next, is the "Specification" phase, where ideas were transformed into graphical visualizations using (UML) class and activity diagrams. Finally, the "Implementation" phase, where the execution plan of the development process was executed, and to better understand how *Crowditivity* works, a guide of the different functionalities that can be used was shown with their respective images taken from the platform itself. In the upcoming chapter, we will talk about the case study to illustrate the use of our platform through a group of users to test the functionalities that were implemented.

Chapter 5

Putting *Crowditivity* to the Test: A Case Study

This chapter will present the case study that was conducted to evaluate user experience and *user experience* (UX) of *Crowditivity*. It covers the methods used to gather results, the series of tasks that were done by the users, and the analysis made on the results. We start by defining the general idea of user experience that is the basis of the questionnaire that was conducted at the end of this study. Then, we define the project statement that was used as an example by the users who participated, and then provide feedback on their experience through the *User Experience Questionnaire Plus* (UEQ+). At the end, we will analyze the results that we got that provided valuable information on the strengths and potential areas of improvement in terms of attractiveness, efficiency, and overall performance of the platform.

5.1 User Experience (UX)

The term ‘user experience’ is associated with a wide variety of meanings [12], ranging from traditional usability to beauty, hedonic, affective or experiential aspects of technology use [16]. A good user experience is a key factor for the market success of products [39]. In general, UX refers to the overall experience that a user has with a product in terms of how easy it is to use, it is everything that touches upon your experience with a product, and in our case a product can be a software, application, website, or any other thing that a user can interact with digitally. It utilizes all aspects of user interactions, such as the user interface, usability, and performance of a product. It is important to understand users’ needs in the development process, this helps to build an efficient and usable product. To better understand the importance of UX, we will take the example of websites on the Internet. We know that we can find a huge number of websites on the internet, and some offer the same products to the user but are created by a different company, so what makes one better than the other in this case? It is in the way users engage with their website, and on that note, we are certain that a good user experience can differentiate a company’s product from its competitors that used a poor design approach, which leads to the overall success of that company. On that matter, designers should manage every aspect of the user experience, they have to focus on creating a product that functions properly, is easy to learn, and easy to use. So, when building a product that is going to be used by users, we have to rely on the idea that not every user is an expert in technology. There are many components to UX that will inform the design to enhance a user experience, like figure 5.1

that shows the UX honeycomb developed by "Peter Morville", famously known as the "founding father" of information architecture [35]. The honeycomb he developed identifies how each UX "facet" is important to answer the user needs given a product technical limitations, and to achieve an optimal level of user experience, an iterative process involves multiple design ideas to allow continuous improvements of the user interface design. By gathering user feedback, we gain insight into the preferences of the user, which can help us create a more user-centered experience that drives to deliver an interface that aligns with the requirements but also improves the overall user satisfaction.



Figure 5.1: UX Honeycomb

5.2 Case Selection

This section provides an explanation on the project that is being used to conduct the case study. To effectively let users engage in our study, we needed to create a real world project scenario that reflects on how the platform could be used by professional people when developing an application. To that extent, we thought of an idea for an application that is still in its early stages of development. This application which we named "*Research Connect*" needs a structured representation for its requirements and has some design flows in its mock-ups that need to be adjusted. The idea of this application revolves around connecting researchers to users, it is like a professional social network, where researchers from any educational field of study are searching for motivated individuals to participate in their surveys, forms, or questionnaires. We chose to go with this idea because in our case we needed individuals to be part of our study findings, so it was inlined with our research topic and the methodologies we used. We further designed the user interface that is going to be used to develop this application, by creating some mock-ups which are used as a "quick and dirty" way of gathering requirements [34] for user interfaces. To design these mockups, we used a tool called *Balsamiq*¹, which is used to create digital

¹<https://balsamiq.com>

sketches for an application in its very early stages of development. The advantage of using such tool is not to build large and fully interactive prototypes, instead developers gain experience and evaluate components of the wireframe directly on a very early version [11] of an application. Figure 5.2 shows the 3 mock-ups screens of the application:

- **Home Screen:** this is where researchers posts their research topics and users can choose to engage in.
- **Feed:** this screens shows a summary of a certain research that a researcher conducted, it gives a general overview on the time it took and users that were involved
- **Profile:** this section is focused on the profile of the researchers, where they can add their experience and general information regarding their field of research

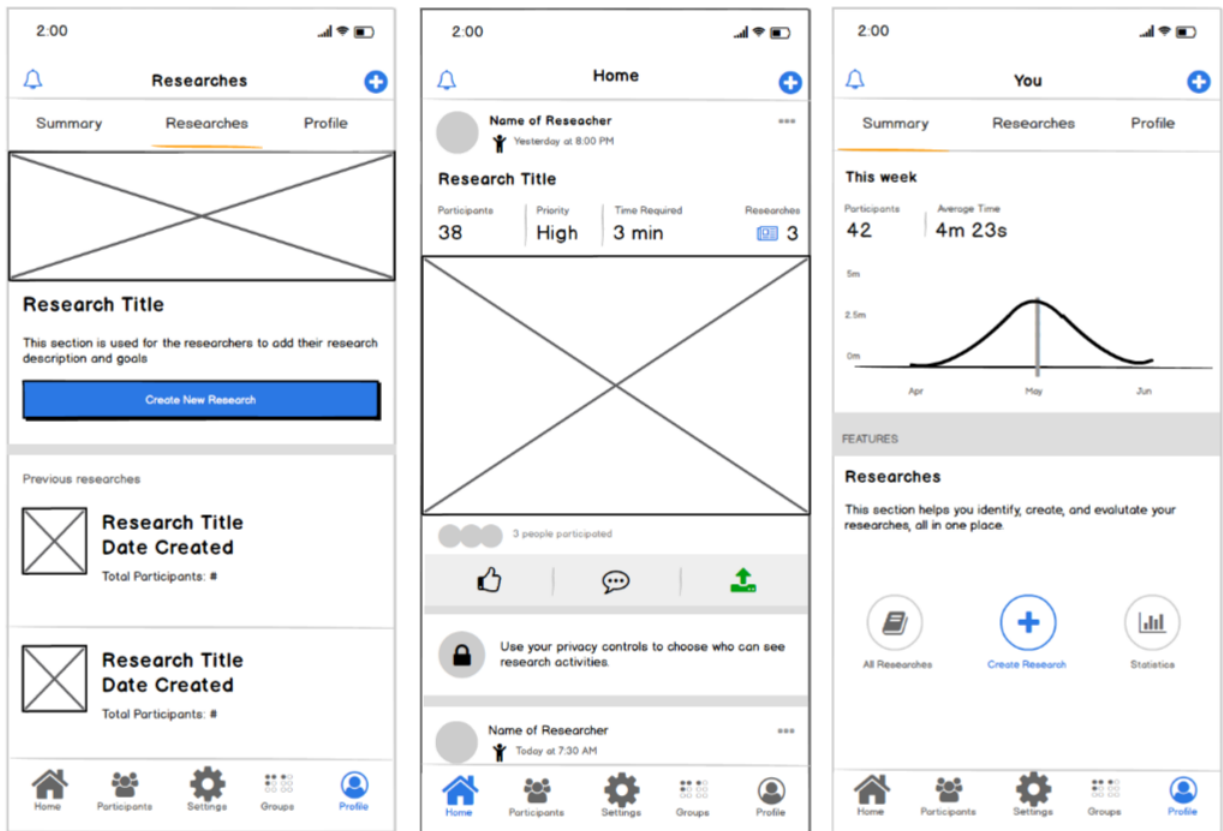


Figure 5.2: Research Connect: User Interface Mock-ups

5.3 Evaluating *Crowditivity*

We defined the project idea for the users in our case study to have a general overview on the purpose of creating requirements and annotate on these design mock-ups. In this section, we will look at how we let users engage in our study and the tasks that were requested from them to perform and evaluate our platform's usability in an effective way. First, we needed to find users to participate in our study; in that case, we asked people to give us a few minutes of their time to explain to them how beneficial they could be if they participated in our case study. Some individuals accepted our proposal and were ready to

start testing our platform. We should mention that the users who participated in our study did not have a technical background, but some of them were quite comfortable with the use of technology. That being said, to let people be comfortable when participating in the case study, we created a step-by-step document in a (.pdf) format and sent it to each individual on their email addresses. The document that we created had an introduction explaining the goal of this case study, the project statement that was mentioned in the previous section, the step by step guide that users needed to follow to test our platform, and at the end, a link to the questionnaire that we created and that will be further discussed in the upcoming section of this chapter. We included a note in our guide that emphasizes the use of computers as the primary testing device to efficiently evaluate all the functionalities of our platform. In the following, we will have an overview of the content found in the document we created:

- **Introduction:** "Hello! Thank you for taking the time to participate in this case study, which is important to us for analyzing the user experience of our platform. Throughout this study, we kindly ask for your participation in performing a series of tasks and sharing your valuable experience on our User Experience Questionnaire (*UEQ+*). Your participation will be focused on testing our platform, ensuring it meets the needs and expectations to perform certain tasks."
- **Project statement:** "We have created some design mock-ups for an application that connects researchers with people who are interested in participating in research studies. This application will facilitate the finding process, making it easier for researchers to find suitable users for their studies."

5.3.1 Steps guide for creating requirements

here we define the steps that are needed to effectively test the platform:

1. "Get started by creating an account"
2. "Verify your email using the link found in your inbox"
3. "Start by creating a new study"
4. "Enter the title for example your: Firstname Lastname"
5. "Enter the following study description: 'Connecting researchers and users'"
6. "Upload the image called "thumbnail" found in the folder we have sent you"
7. "Read the description and click on the Start button to initiate the data collection process"
8. "Enter your demographic information and proceed to the next step"
9. "Upload the image called "home" and click 'Next'"
10. "Now for this step, add the details for each of the following fields"

Requirement Title	Create New Study
Requirement Type	Functional Requirement
Event #	1
Use Case #	1
Description	The application must provide researchers with the ability to create a study
Rationale	Researchers need to effectively communicate their study details to attract potential users
Originator	Vick Lampron, Dev team
Fit criterion	The application should enable researchers to enter study information and description
Customer Satisfaction	5
Customer Dissatisfaction	5
Priority	Crucial
Conflicts	0
Supporting Materials	Study creation UI design, study requirement guidelines
History	19-05-23

11. "Once you are done, click the “Submit” button"
12. "Now, click on the “My Studies” button to check if your study is showing"
13. "Click the “View requirements” button to see the requirement you created in that study"

5.3.2 Steps guide to annotate images

1. "Click on the "Create Annotation" button"
2. "Click the “Add Image” button and choose the image called 'feed'"
3. "Enter the name of the image, for example: Feed Page Modifications"
4. "Click the button “View types” to check all the available annotation types that can be drawn on an image"
5. "Start by adding the 4 drawings shown below ordered from 1 to 4: (select, move down, check, delete), wait for 1 second after adding another types", figure 5.3 shows the example of the annotated image in its before and after states:

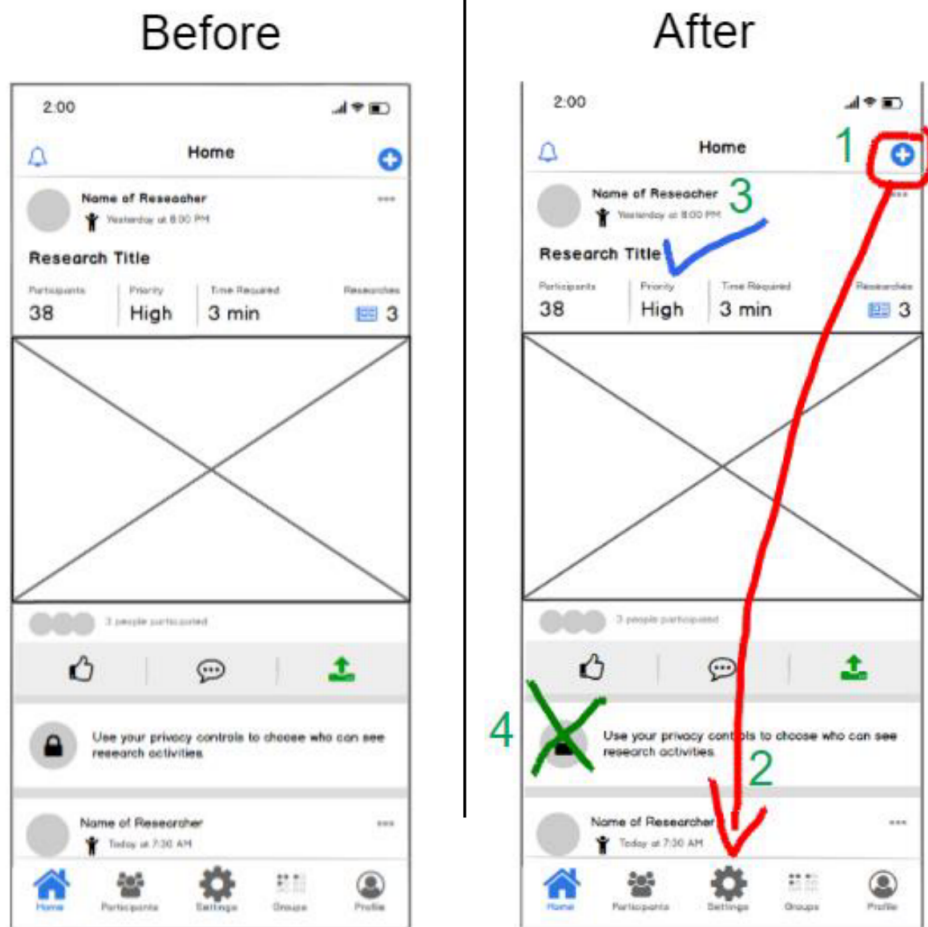


Figure 5.3: Image Annotation Example

6. "After adding the drawings, a list of annotations should appear on the right side of your screen that looks like this" as in figure 5.4

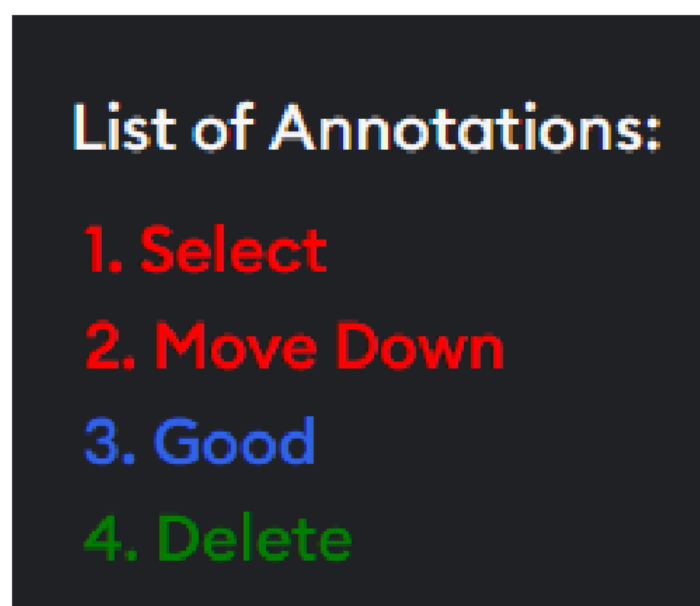


Figure 5.4: List of Recognized Annotations

7. "Click the button 'Upload Image' to upload your image to the database"
8. "Check that your annotation was successfully created in the 'My Annotations' page"

- **Final Step:** "Click the "Log out" button to conclude your testing session"

Upon letting users participate in this case study, we will give a quick overview of what the user actually had to do and understand the purpose behind it. Since we thoroughly discussed the implementation and aims behind *Crowditivity*, we created this process to focus on testing all its functionalities. In our project statement we mentioned the idea of an application that needed requirements and some design changes, and with that in mind, we can make sure that we are testing the overall usability, performance, and how users were able to engage without any prior knowledge of our platform. First, we requested the user to authenticate; this includes creating a new account and verifying their email addresses. Then, they were requested to create a study to elicit requirements for this application that is in its early stages of development called "Research Connect". After they created a study, they added their demographic information such as: (first name, last name, age, nationality, gender, and education), then they chose an image that can have its own set of requirements and add the proper details in the desired fields. Once all the fields have been completed, the users submit their requirements to our database, and to make sure that everything was successful, a page with their studies and requirements is shown at the end of this evaluation. This was the first functionality of our platform that was tested. Secondly, when the user completed the steps required to create requirements for a study, they were requested to add an image of a mock-up that was designed specifically for the user interface of this application. Users were able to add certain annotations from the set of predefined types identified in Chapter 3. We used types such as: "Select, Move Bottom, Check, and Delete" that were randomly chosen to make sure that we test different annotations rather than the ones that only had the best agreement rates. Table 5.1 provides statistical information on the total number of annotations created by all participants, indicating how many times the annotations drawn were recognized with their respective types, based on the participants image annotations that can be found in the Appendix section **User Annotations from Case Study**. After the users have successfully completed all the aforementioned steps, the last thing to do in our study was to ask them to participate in a questionnaire aimed at gathering their feedback on the user experience of our platform. The questionnaire will be discussed furthermore in details in the next section.

Annotation Type	Number of Annotations
Select	9
Move Down	8
Good	7
Delete	8
Line	9
Move Up	1
Rotate Left	1
Rotate Right	1
Total	44

Table 5.1: Table of Annotations: Distribution of Types and Total Number of Annotations

5.4 UEQ+

5.4.1 Defining *UEQ+*

In the following section, we will introduce the User Experience Questionnaire Plus (*UEQ+*) that users participated in to evaluate the user experience of our platform. *UEQ+* is a modular extension of the User Experience Questionnaire (*UEQ*) which is an end user questionnaire to measure user experience quickly in a simple and immediate way while covering a preferably comprehensive impression of the user experience of the product [27]. The *UEQ* was originally developed in German and English. As a modular extension, the *UEQ+* has more user experience (UX) scales compared to the *UEQ* and can be used to evaluate products in special scenarios [37]. *UEQ+* is designed to gather feedback from users about their subjective experience. In addition, the effort to set up a *UEQ+* for a project can also create a deeper common understanding of the UX requirements and their importance in the project team and can therefore create some additional value [18]. The framework contains several scales that measure different UX aspects. These scales can be combined to cover relevant research questions [40]. We chose to measure it using these 9 scales, defined on the *UEQ+*'s website as:

- **Attractiveness:** Overall impression of the product. Do users like or dislike it?
- **Efficiency:** Can users solve their tasks without unnecessary effort? Does it react fast?
- **Stimulation:** Is it exciting and motivating to use the product? Is it fun to use?
- **Novelty:** Is the design of the product creative? Does it catch the interest of users?
- **Usefulness:** Does using the product bring advantages?
- **Value:** Does the product design look professional and of high quality?
- **Visual Aesthetics:** Does the product look beautiful and appealing?
- **Intuitive Use:** Can the product be used immediately without any training or help?
- **Clarity:** Impression towards order, structure and visual complexity of a graphical user interface

To have a more detailed description of *UEQ+* and its usage scenarios, you can look up *UEQ+* handbook [38].

5.4.2 The Questionnaire

We invited users to participate in the *UEQ+* assessment that we created, and the idea behind their participation is that their feedback will help us improve the overall usability and will contribute in improving *Crowditivity*. For ease of use, we added the link at the end of the our step by step guide that would take the users directly to the form where they can add their feedback. We created the questionnaire using *Google Forms*, an online forms and surveys creator with multiple question types, this tool will help us analyze the results of our *UEQ+* questionnaire in an efficient way. The questionnaire included questions using a 7-point Likert scale [29]. An example of this questionnaire can be found

in section [Google Forms](#). We started by giving an introduction that explains what are the goals of this questionnaire:

- "Hello! Your valuable input plays a crucial role in enhancing the platform's usability and improving the overall research experience. Please complete the following questionnaire to assess the product. The questionnaire contains opposing pairs of product properties. The grades between the opposites are indicated by circles. Check one of the circles to indicate your level of agreement with the individual terms. Thank you for being part of our questionnaire!"

We asked general questions to gather information about the users:

1. "What is your age?"
2. "What is your gender?"
3. "I use a smartphone frequently on a scale from 1 to 7?"
4. "I use a computer frequently on a scale from 1 to 7?"
5. "I use a tablet frequently on a scale from 1 to 7?"

Below are the questions that were asked based on the 9 scales that we mentioned earlier:

- **Attractiveness:**

In my opinion, the product is generally

bad	☐	☐	☐	☐	☐	☐	☐	good
unfriendly	☐	☐	☐	☐	☐	☐	☐	friendly
	1	2	3	4	5	6	7	

- **Efficiency:**

To achieve my goals, I consider the product as

slow	☐	☐	☐	☐	☐	☐	☐	fast
impractical	☐	☐	☐	☐	☐	☐	☐	practical
	1	2	3	4	5	6	7	

- **Stimulation:**

In my opinion, handling and working with the product are

boring	☐	☐	☐	☐	☐	☐	☐	exiting
not interested	☐	☐	☐	☐	☐	☐	☐	interested
	1	2	3	4	5	6	7	

- **Novelty:**

In my opinion, the idea behind the product and its design are

dull	☐	☐	☐	☐	☐	☐	☐	creative
conservative	☐	☐	☐	☐	☐	☐	☐	innovative
	1	2	3	4	5	6	7	

- **Usefulness:**

I consider the possibility of using the product as

useless	☐	☐	☐	☐	☐	☐	☐	useful
not beneficial	☐	☐	☐	☐	☐	☐	☐	beneficial
	1	2	3	4	5	6	7	

- **Value:**

I generally consider the design of the product as

not elegant	☐	☐	☐	☐	☐	☐	☐	elegant
not presentable	☐	☐	☐	☐	☐	☐	☐	presentable
	1	2	3	4	5	6	7	

- **Visual Aesthetics:**

In my opinion, the visual design of the product is

ugly	☐	☐	☐	☐	☐	☐	☐	beautiful
unappealing	☐	☐	☐	☐	☐	☐	☐	appealing
	1	2	3	4	5	6	7	

- **Intuitive Use:**

In my opinion, using the product is

difficult	☐	☐	☐	☐	☐	☐	☐	easy
illogical	☐	☐	☐	☐	☐	☐	☐	logical
	1	2	3	4	5	6	7	

- **Clarity:**

In my opinion the user interface of the product looks

unstructured	☐	☐	☐	☐	☐	☐	☐	structured
poorly grouped	☐	☐	☐	☐	☐	☐	☐	well grouped
	1	2	3	4	5	6	7	

5.4.3 Results of the *UEQ+*

Based on the feedback that was received from this questionnaire, we were able to gather valuable results that are going to be analyzed in this section (see the results in table 5.3). And to efficiently do that, we used the *UEQ+ Data Analysis Tool* authored by Dr. Martin Schrepp and can be downloaded here ². This tool has worksheets that are useful to calculate the following: Mean, Variance, Standard deviation, Confidence, and Confidence Interval for each scale found in the questionnaire:

- **Mean:** is the sum of all values in the dataset divided by the total number of values
- **Variance:** Variance is a measure of how spread out the values in a dataset are from the mean we calculated
- **Standard Deviation:** is the square root of the variance. It provides a measure of the spread of the data points around the mean
- **Number of Participants (N):** is the square root of the variance. It provides a measure of the spread of the data points around the mean
- **Confidence:** refers to the level of certainty associated with an estimate. It is often expressed as a percentage, such as 95% confidence level.
- **Confidence Interval:** is a range of values that is likely to contain the true parameter with a certain level of confidence.

²https://ueqplus.ueq-research.org/Material/UEQ_Plus_Data_Analysis_Tool.xlsx

Information on Participants

ID	Age	Gender	Smartphone	PC	Tablet
1	37	Female	7	7	1
2	22	Male	7	7	3
3	20	Male	7	7	7
4	19	Male	7	6	1
5	30	Male	7	7	6
6	20	Male	7	7	3
7	29	Male	7	7	1
8	25	Male	7	7	1

Table 5.2: Demographics of the 8 Participants in the Case Study

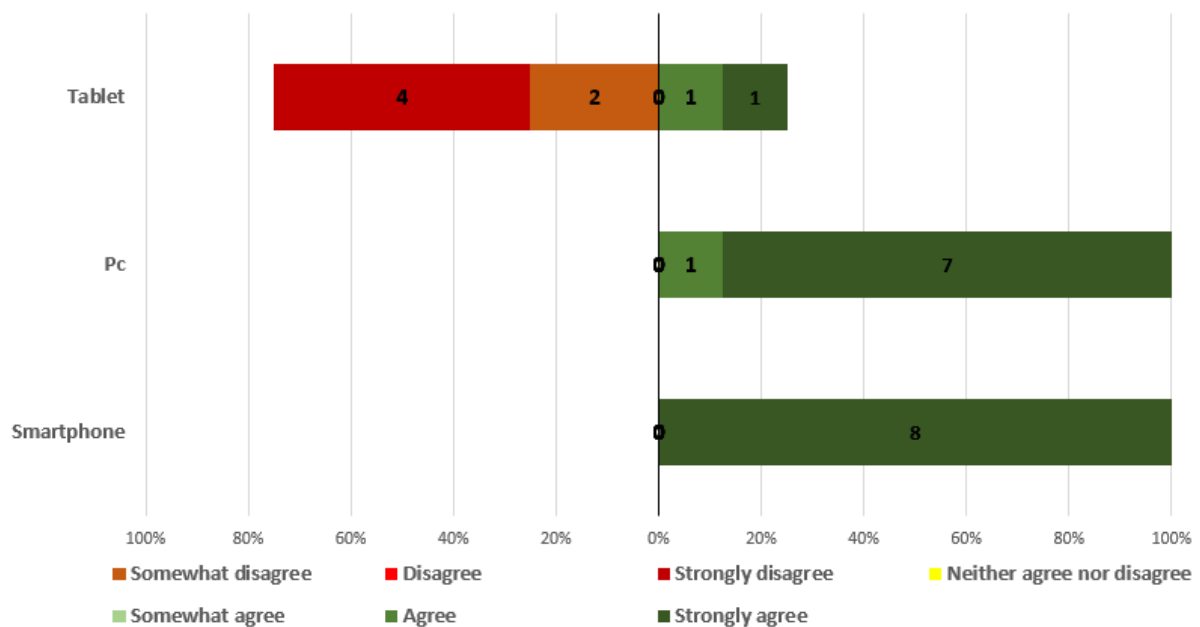


Figure 5.5: Likert Chart on Agreement Scaling - *UEQ+*

Analyzing the Results

Table 5.3 shows the calculations of the different measures that we mentioned earlier. Originally, the scores were on a scale from 1 to 7, but to match the format used in the original version of the *UEQ* analysis tool adjusted them to a scale from **-3** to **+3**. Using this change, the results can be compared more easily across different versions of the questionnaire.

Scale	Mean	Variance	Std.dev.	N	Confidence	Confidence Interval	
Attractiveness	2.81	0.29	0.53	32	0.18	2.63	2.99
Efficiency	2.56	0.53	0.70	32	0.24	2.32	2.81
Stimulation	2.44	0.80	0.86	32	0.30	2.14	2.74
Novelty	2.81	0.16	0.39	32	0.14	2.68	2.95
Usefulness	2.44	1.20	1.06	32	0.37	2.07	2.80
Value	2.81	0.16	0.39	32	0.14	2.68	2.95
Visual Aesthetics	2.81	0.16	0.39	32	0.14	2.68	2.95
Intuitive Use	2.69	0.36	0.58	32	0.20	2.49	2.89
Clarity	2.69	0.23	0.46	32	0.16	2.53	2.85

Table 5.3: Mean, Variance, Standard Deviation, Confidence, and Confidence Interval per Scale Calculated using *UEQ+*'s Analysis Tool

In figure 5.6 shown below, we can see that each bar on the chart represents a different scale from the 9 scales that were used. In the chart, the length of the bar corresponds to the mean score of a specific scale, which is represented in a range from -3 to +3. The higher the mean values the higher the length of the bar is, this indicates a more positive outcome. We need to understand the importance of these visualizations, they provide an easy-to-understand representation of the results collected from the questionnaire, and by looking at the chart, we can quickly interpret which areas of the user experience are strong and which might need some improvements. *UEQ+* has different scales, each made up of four questions, however, in our study we only used two questions from each scale. Eight participants that took part in our study, giving us a total of 16 responses per scale. If we had use all four questions per category, we would have had 32 responses, so to compare our results with other studies that used all four questions per category, we multiplied the number of participants (N) by four, resulting in N=32 to get results that are correct.

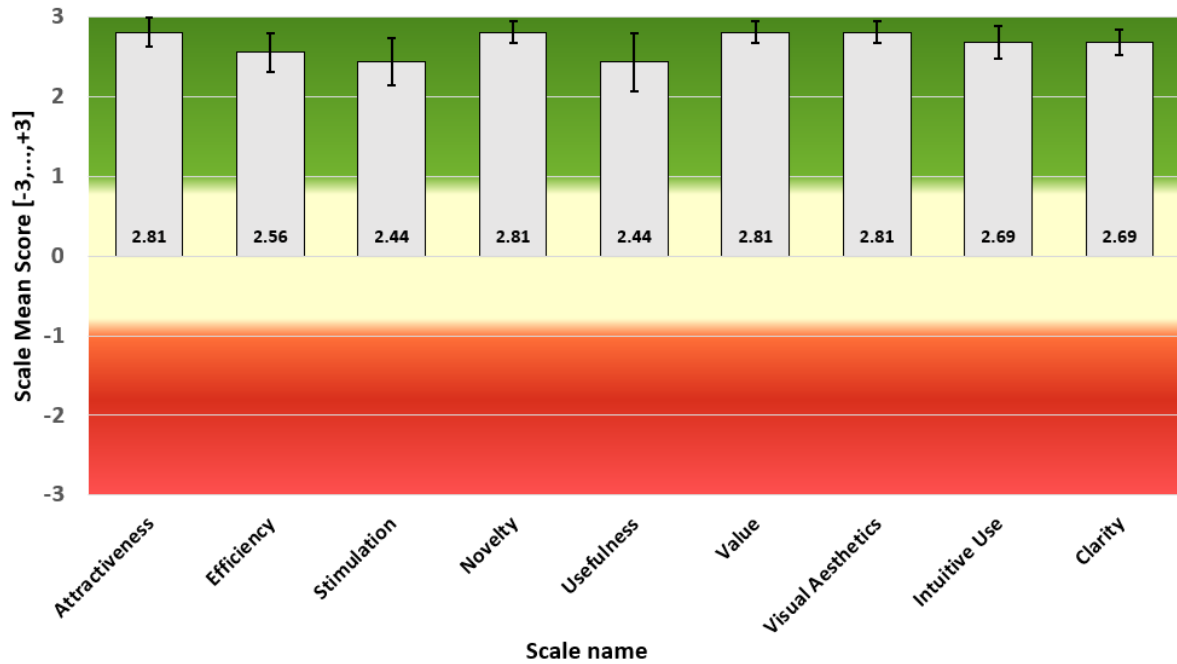


Figure 5.6: Graphical Representation of the Results using *UEQ+*'s Analysis Tool

Discussion about the Results

The feedback we received from the questionnaire showed that all the scales on *UEQ+* have shown positive results, with mean values close to 3 on a scale of -3 to +3. This indicates that users have given favorable responses across the different aspects of the user experience.

We start by the highest values that we got from the following scales: "Attractiveness", "Novelty", "Value", and "Visual Aesthetics" that showed mean scores of **2.81**, these scores indicate that users got a positive outcome, enjoyed its visual design and got value from the tasks they performed on our platform. The low variance and standard deviation in these categories further suggest a strong consensus on these scales.

Then, "Clarity" and "Intuitive Use" have mean scores of around **2.69**, indicating that users found the platform to be well structured and easy to use. These values show that *Crowditivity* balanced the user's needs with innovative design.

Finally, "Efficiency", "Stimulation", and "Usefulness" have slightly lower mean scores of **2.56** and **2.44**, this shows a potential room for some improvements regarding the user experience in these areas. Nonetheless, these scores are considered quite high and some enhancements can be done to the usefulness of tasks performed, this will make our platform more engaging and could enhance the user satisfaction in the future.

In order to reach a certain level of optimality in terms of user experience, we need to maximize the scores by implementing some minor changes that users identify as valuable; this would be beneficial to improve its usability even more. Having the opportunity to get feedback from users was very important for us to identify the strengths and weaknesses of *Crowditivity*. Nevertheless, *Crowditivity* excelled in many key areas, and we might need to address some of its shortcomings when it comes to the "Efficiency" and "Stimulation" by implementing changes related to this scale in the future.

5.5 Conclusion

The case study that we conducted showed how important it is to make a platform easy and pleasant to use. After asking users to test and participate in our *UEQ+* we had an idea of how they felt about using *Crowditivity*. The results that we got from this questionnaire showed that it is really important to pay attention to the UX to keep a platform successful because it evaluated the users' experiences in a quantifiable manner which helped us identify various aspects of the UX, from the simplicity of its navigation to the overall performance and usability. To conclude this chapter, the most important thing that we wanted for our platform is to be as user-friendly as possible, and the results from this case study showed that we have made promising progress towards this goal.

Chapter 6

Conclusion and Future Work

6.1 Review

In this chapter, we will review the progress made in this thesis by discussing the steps taken to achieve our results:

First, in **Chapter 2: Related Work**, we discussed about the user interface requirements elicitation and its importance in our master thesis. We also mentioned different topics such as SDLC and requirements engineering for user interface. Then, we explored the previous work that was done in terms of elicitation studies, where we saw platforms such as the Crowdesign engine used to efficiently elicit requirements and Gelicit for gathering gesture requirements. We also mentioned tools like *VOLERE*'s SHELL template, ReDSeeDS, and other tools as well that are useful for creating structured requirements.

Then, in **Chapter 3: Eliciting and Identifying Annotation Types**, our focus in this chapter was to identify useful annotation types. We conducted a crowd elicitation study to let participants create their own annotation types on digital canvases. Then, we used the results to conduct a survey to identify which annotation types are best to be used. Along that, we classified the different types of annotation, which helped us calculate the degree of consensus between participants using *AGATe* [49]. Overall, this chapter provided information about the methodology used to create annotation types which are going to be used and recognized in our platform.

Afterwards, in **Chapter 4: Development of *Crowditivity***, we gave a detailed explanation of the development of *Crowditivity*, by explaining its conception, requirements, architecture, and implementation. We also mentioned how we integrated the different types of annotation identified in Chapter 3 and the use of the *\$P+* algorithm to recognize these types.

Finally, in **Chapter 5: Putting *Crowditivity* to the Test: A Case Study**, we conducted a case study to evaluate the user experience of *Crowditivity*. We presented the participants with a series of tasks to test our platform using an example of a real-world scenario, then they provided feedback on their experience using the *UEQ+* method for assessment, which allowed us to gather qualitative data on their experience.

6.2 Critical Analysis

This master thesis presented a comprehensive study on *Crowditivity*'s development, implementation, and user evaluation. It is worth noting that the case study showcased an overall positive user feedback, and although the platform scored highly in terms of visual, the lower scores found in scales such as "Stimulation" and "Usefulness" indicate potential areas of improvements. It would be beneficial to investigate these aspects in depth to better understand the challenges faced by users and how our platform can evolve to address them. As for the additional features that could be added in *Crowditivity*, it would be worth exploring those that aim at improving the overall user experience of the platform, these could include personalized user interfaces or guided tutorials for easier navigation. In conclusion, *Crowditivity* has demonstrated its potential as an innovative and user-friendly platform, positive user feedback attests to its robust functionalities that showcase its success in balancing user needs with technological advances.

6.3 Future Work

Although our work for this thesis has reached its end, there are new features that can be implemented in *Crowditivity* in the short-term as well as the mid-term:

6.3.1 Short-term

Short-term features refer to near-future additions to *Crowditivity*. These are simpler and prioritized enhancements that be implemented in less time and fewer resources.

- **Expand Annotation Types.** We can explore more annotation types that can be added to our predefined set; this would help users have more flexibility to express their requirements in a more comprehensive way. By adding new annotation types we would extend their recognition capabilities in *Crowditivity*, allowing it to interpret newly added annotations types.
- **User-Defined Recognition.** This feature encourages users to add their own custom annotation types and train their recognition using $SP+$. This would provide users with the flexibility to use their own annotation types that are unique to their specific requirements. By giving the freedom for our users to create more personalized types we improve our platform's usability.

6.3.2 Mid-term

Mid-term features are improvements planned over a longer period of time, these entail a more advanced development effort, including comprehensive research, extensive user testing, and may involve the use of advanced technologies.

- **Digital Ink Annotations.** This feature allows users to create annotations using *digital ink* which is the digital representation of a drawing created using a stylus or digital pen on a digital device. This would provide a more natural way for users to interact with our platform.

- **Analyze *VOLERE* Requirements.** This feature automatically analyzes the requirements, while extracting key information, and generating a summarized document for these requirements. This process would reduce the manual effort and improve efficiency to produce high-quality documents.

List of Figures

2.1	Software Development Life Cycle	8
2.2	Stakeholders' perspective on Requirements	10
2.3	Crowdesign Engine	13
2.4	Study Information	14
2.5	Referent Choice	14
2.6	Symbol Type	14
2.7	User Feedback	14
2.8	Study Summary	14
2.9	Similar Symbols	15
2.10	Agreement Rate	15
2.11	Real World Example	16
2.12	Experts vs Non-Experts	16
2.13	Voting Matrix	17
2.14	SHELL Template	19
2.15	V-Model	19
2.16	ReqView Template	20
2.17	Traceability Report	20
2.18	Workflow Steps	21
2.19	Gelicit's Meta-model	23
2.20	6 Steps Workflow of GES in Gelicit [30]	25
2.21	Gesture acquisition: (a) main area, (b) 2D1/2 object tracking, (c) touch + (d) air gesture, (e) 3D gesture [30]	26
2.22	Gelicit Technical Parameters [30]	27
2.23	Model-Based Engineering	28
2.24	Overview of the ReDSeeDS Tool	30
2.25	SnappView Screenshot Example	31
3.1	Example of annotation types [44]	34
3.2	Continued [44]	34
3.3	Example of a question on forms.app	35
3.4	Age Distribution of Participants in the Elicitation of Annotation Types Study	39
3.5	Likert Chart on Agreement Scaling - Eliciting Annotation Types	39
3.6	Classification of annotation types	41
3.7	Elicited Results for the Annotation of Type "Attention"	42
3.8	Agreement Rate Formula [49]	43
3.9	Referent Agreement Rates calculated on <i>AGATe</i> [49] using the Classification Data of Table 3.2	44
3.10	Agreement rates that were computed by <i>AGATe</i> [49] for referents, which are sorted in decreasing order, with 95% confidence intervals	46

3.11	Age Distribution of Participants in the Survey of Identifying Annotation Types	52
3.12	Likert Chart on Agreement Scaling - Identifying Annotation Types	52
3.13	Identification Survey Results: An Overview of the Identified Annotation Types	53
4.1	Landing Page - <i>Crowditivity</i>	56
4.2	Conceptual Diagram - <i>Crowditivity</i>	60
4.3	Logical Diagram - <i>Crowditivity</i>	61
4.4	Main Activity Diagram - <i>Crowditivity</i>	62
4.5	Study Activity Diagram - <i>Crowditivity</i>	63
4.6	Requirements Activity Diagram - <i>Crowditivity</i>	64
4.7	Annotations Activity Diagram - <i>Crowditivity</i>	65
4.8	Admin Activity Diagram - <i>Crowditivity</i>	66
4.9	Technology Stack Diagram - <i>Crowditivity</i>	67
4.10	Sign in Page - <i>Crowditivity</i>	69
4.11	Sign up Page - <i>Crowditivity</i>	70
4.12	Create Study - <i>Crowditivity</i>	70
4.13	Study Description - <i>Crowditivity</i>	71
4.14	Personal Information - <i>Crowditivity</i>	71
4.15	Choosing Images - <i>Crowditivity</i>	72
4.16	<i>VOLERE</i> Template - <i>Crowditivity</i>	72
4.17	Complete - <i>Crowditivity</i>	73
4.18	Add an Image - <i>Crowditivity</i>	73
4.19	Annotate Image - <i>Crowditivity</i>	74
4.20	Gesture articulations produced by people with low vision present more variation than gestures produced by people without visual impairments, which negatively affects recognizers' accuracy rates. Ten (10) superimposed executions of a "star" gesture are shown for six people: a person without visual impairments (a); three people with congenital nystagmus and high myopia (b), (c), (d); and two people with chorioretinal degeneration (e), (f). [47]	75
4.21	Annotation Types used for Recognition in <i>Crowditivity</i>	76
5.1	UX Honeycomb	79
5.2	Research Connect: User Interface Mock-ups	80
5.3	Image Annotation Example	83
5.4	List of Recognized Annotations	83
5.5	Likert Chart on Agreement Scaling - <i>UEQ+</i>	88
5.6	Graphical Representation of the Results using <i>UEQ+</i> 's Analysis Tool . . .	89
B.1	Participant N°1	100
B.2	Participant N°2	100
B.3	Participant N°3	101
B.4	Participant N°4	101
B.5	Participant N°5	101
B.6	Participant N°6	102
B.7	Participant N°7	102
B.8	Participant N°8	102

B.9 Participant N°9	103
B.10 Participant N°10	103
B.11 Participant N°11	103
B.12 Participant N°12	104
B.13 Participant N°13	104
B.14 Participant N°14	104
B.15 Participant N°15	105
B.16 Participant N°16	105
B.17 Participant N°17	105
B.18 Participant N°18	106
B.19 Participant N°19	106
B.20 Participant N°20	106
B.21 Participant N°21	107
B.22 Participant N°22	107
B.23 Participant N°23	107
B.24 Participant N°24	108
B.25 Participant N°25	108
B.26 Participant N°26	108
C.1 Annotation Type: Add	109
C.2 Annotation Type: Attention	110
C.3 Annotation Type: Bad	110
C.4 Annotation Type: Confusion	111
C.5 Annotation Type: Delete	111
C.6 Annotation Type: Good	112
C.7 Annotation Type: Line	112
C.8 Annotation Type: Move Bottom	112
C.9 Annotation Type: Move Top	113
C.10 Annotation Type: Move Left	113
C.11 Annotation Type: Move Right	113
C.12 Annotation Type: Optional	114
C.13 Annotation Type: Required	114
C.14 Annotation Type: Rotate Left	114
C.15 Annotation Type: Rotate Right	115
C.16 Annotation Type: Select	115
C.17 Annotation Type: Zoom-in	115
C.18 Annotation Type: Zoom-in	116
D.1 Annotation Type Recognition: Add	117
D.2 Annotation Type Recognition: Attention	118
D.3 Annotation Type Recognition: Bad	118
D.4 Annotation Type Recognition: Confusion	119
D.5 Annotation Type Recognition: Delete	119
D.6 Annotation Type Recognition: Good	120
D.7 Annotation Type Recognition: Line	120
D.8 Annotation Type Recognition: Move Bottom	121
D.9 Annotation Type Recognition: Move Top	121
D.10 Annotation Type Recognition: Move Left	122
D.11 Annotation Type Recognition: Move Right	122

D.12 Annotation Type Recognition: Optional	123
D.13 Annotation Type Recognition: Required	123
D.14 Annotation Type Recognition: Rotate Left	124
D.15 Annotation Type Recognition: Rotate Right	124
D.16 Annotation Type Recognition: Select	125
D.17 Annotation Type Recognition: Zoom-in	125
D.18 Annotation Type Recognition: Zoom-in	126
E.1 Question Pro - Identification Survey: Introduction	127
E.2 Question Pro - Identification Survey: Demographic Information	128
E.3 Question Pro - Identification Survey: Questions I	128
E.4 Question Pro - Identification Survey: Questions II	129
E.5 Question Pro - Identification Survey: Questions III	129
E.6 Forms.app - Elicitation Study: Introduction	130
E.7 Forms.app - Elicitation Study: Demographic Information	131
E.8 Forms.app - Elicitation Study: Questions I	132
E.9 Forms.app - Elicitation Study: Questions II	133
E.10 Forms.app - Elicitation Study: Questions III	134
E.11 Google Forms - UEQ+: Introduction	135
E.12 Google Forms - UEQ+: Demographic Information	136
E.13 Google Forms - UEQ+: Questions I	137
E.14 Google Forms - UEQ+: Questions II	138
E.15 Google Forms - UEQ+: Questions III	139
G.1 Image Annotations Participant 1	147
G.2 Image Annotations Participant 2	148
G.3 Image Annotations Participant 3	148
G.4 Image Annotations Participant 4	149
G.5 Image Annotations Participant 5	149
G.6 Image Annotations Participant 6	150
G.7 Image Annotations Participant 7	150
G.8 Image Annotations Participant 8	151

List of Tables

3.1	Participant Demographics in the Elicitation Study: Sample size of 26 Participants	38
3.2	Classification of Elicited Annotation Types	42
3.3	Referents and their Classification Groups	43
3.4	Referent Agreement Rates calculated on <i>AGATE</i> [49]	45
3.5	Participant Demographics in the Identification Survey: Sample Size of 59 Participants	50
3.6	Continued. Participant Demographics in the Identification Survey: Sample Size of 59 Participants	51
3.7	Participant Responses: Percentages of Chosen Options per Question	53
5.1	Table of Annotations: Distribution of Types and Total Number of Annotations	84
5.2	Demographics of the 8 Participants in the Case Study	88
5.3	Mean, Variance, Standard Deviation, Confidence, and Confidence Interval per Scale Calculated using <i>UEQ+</i> 's Analysis Tool	89

Appendix A

General Note

ChatGPT, an Artificial Intelligence (AI) tool, has been used to assist in text revision and correction to improve the readability within the context of this master thesis. All other sources that were also used in the work of this thesis have been clearly cited.

Appendix B

Results of the Elicitation Study on Annotation Types

This appendix regroupes the results elicited in Chapter 3

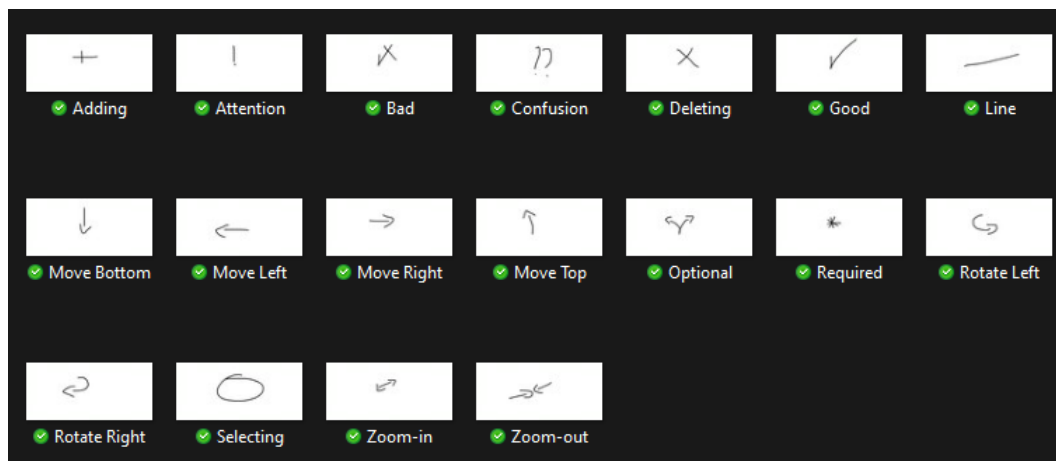


Figure B.1: Participant N°1

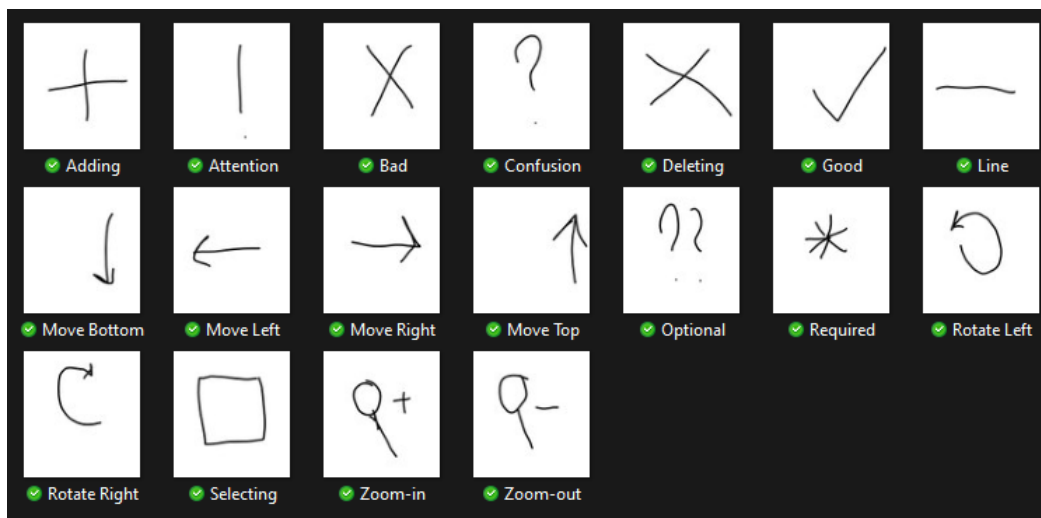


Figure B.2: Participant N°2

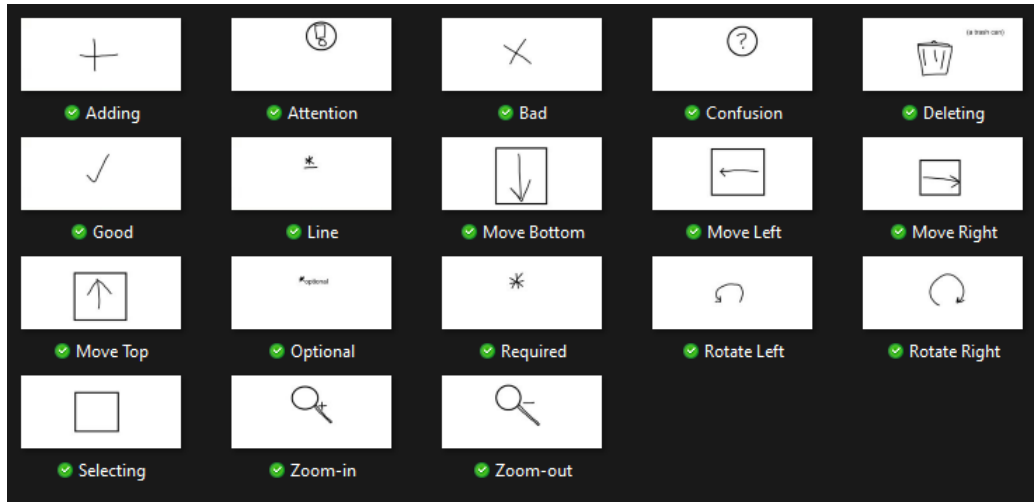


Figure B.3: Participant N°3

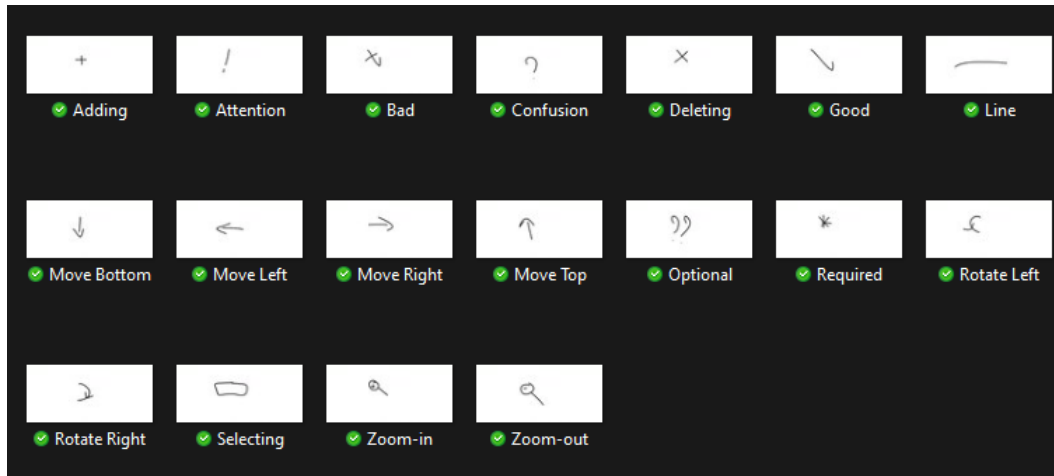


Figure B.4: Participant N°4

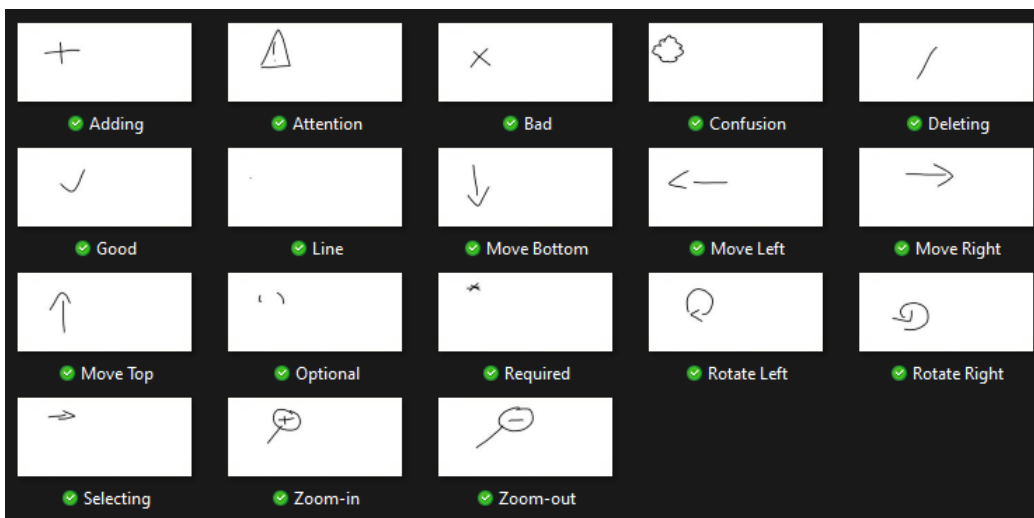


Figure B.5: Participant N°5

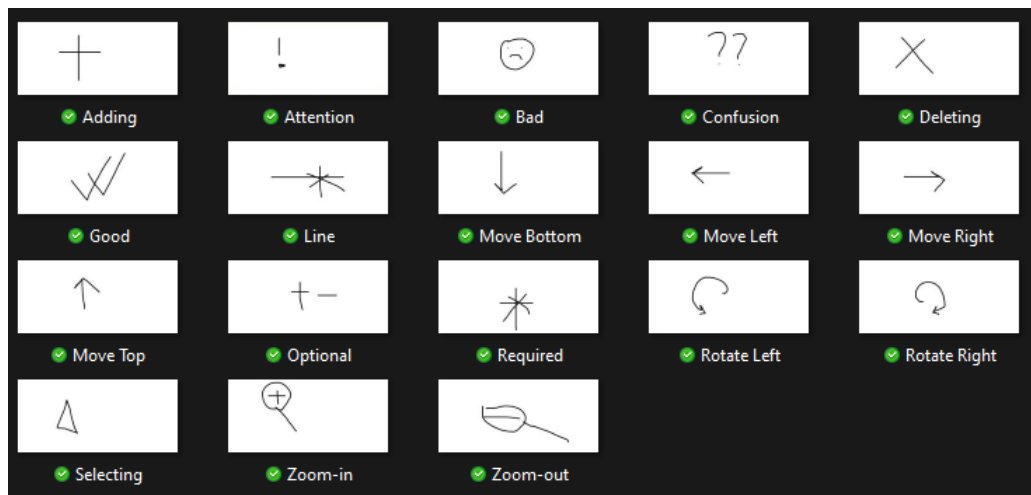


Figure B.6: Participant N°6

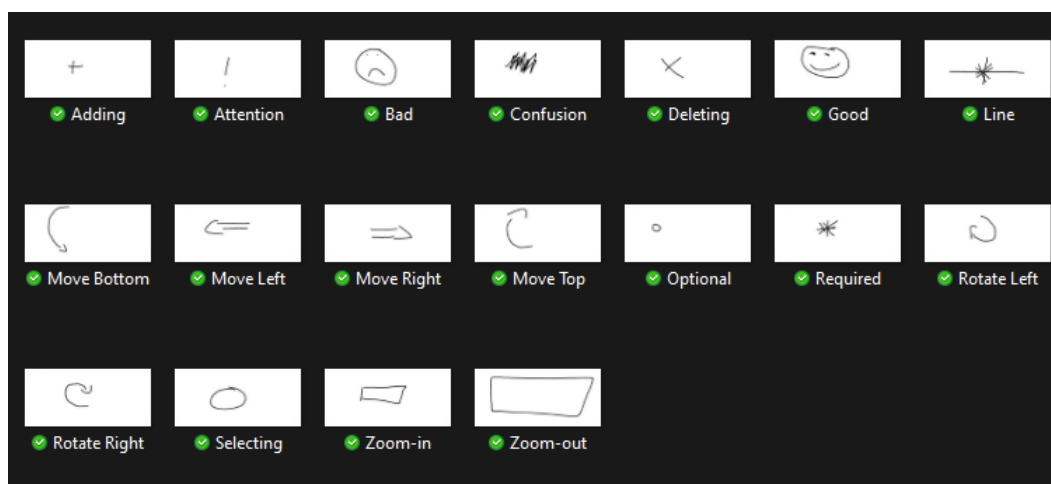


Figure B.7: Participant N°7

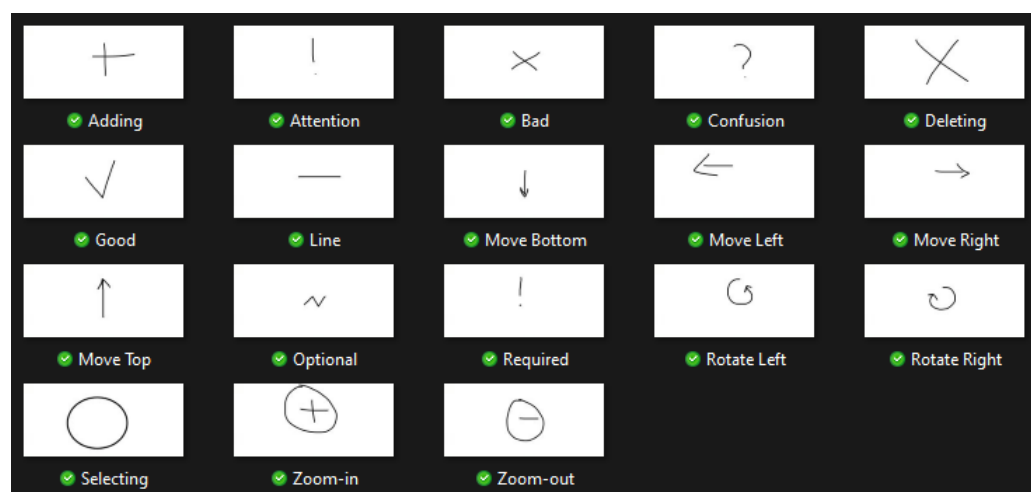


Figure B.8: Participant N°8

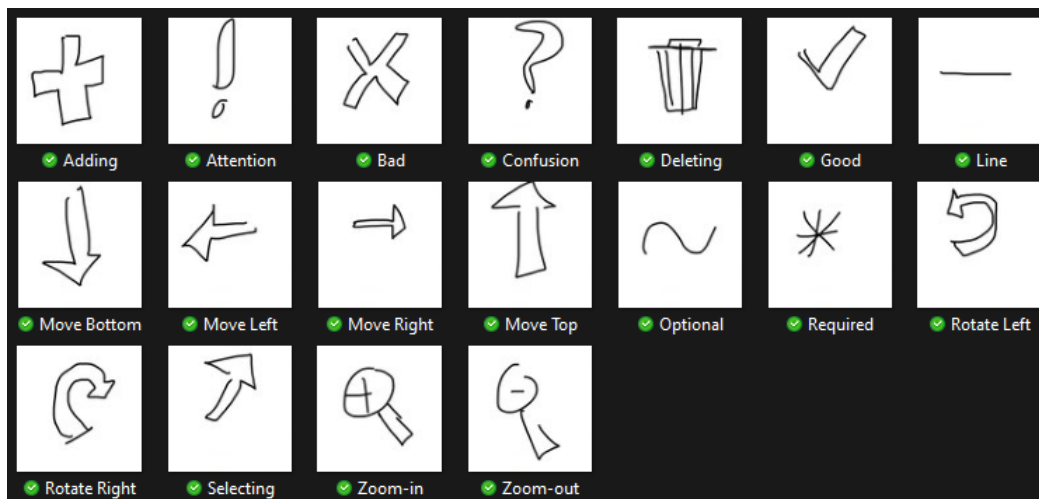


Figure B.9: Participant N°9

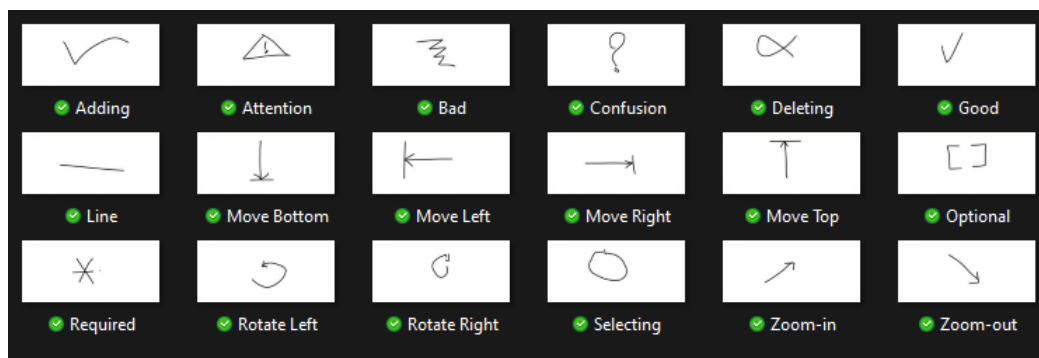


Figure B.10: Participant N°10

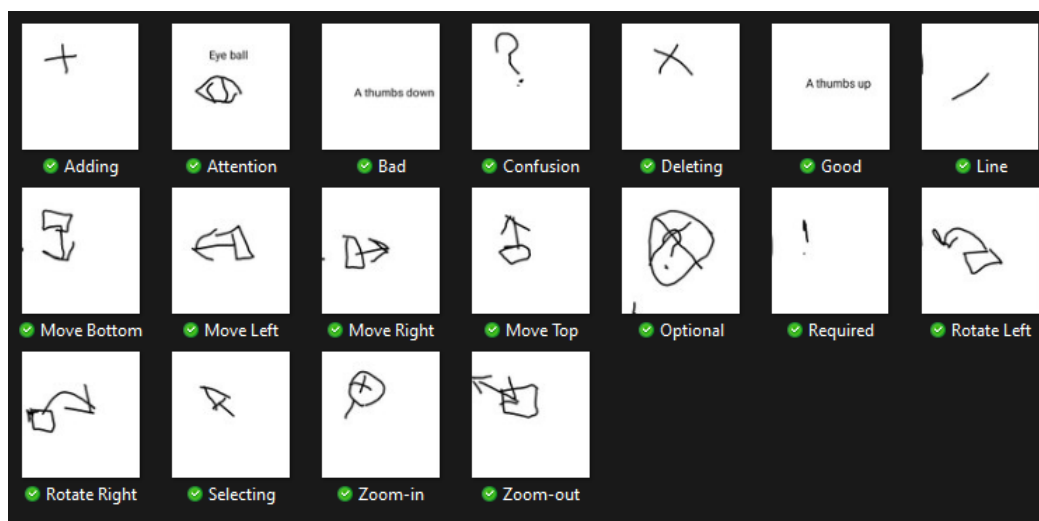


Figure B.11: Participant N°11

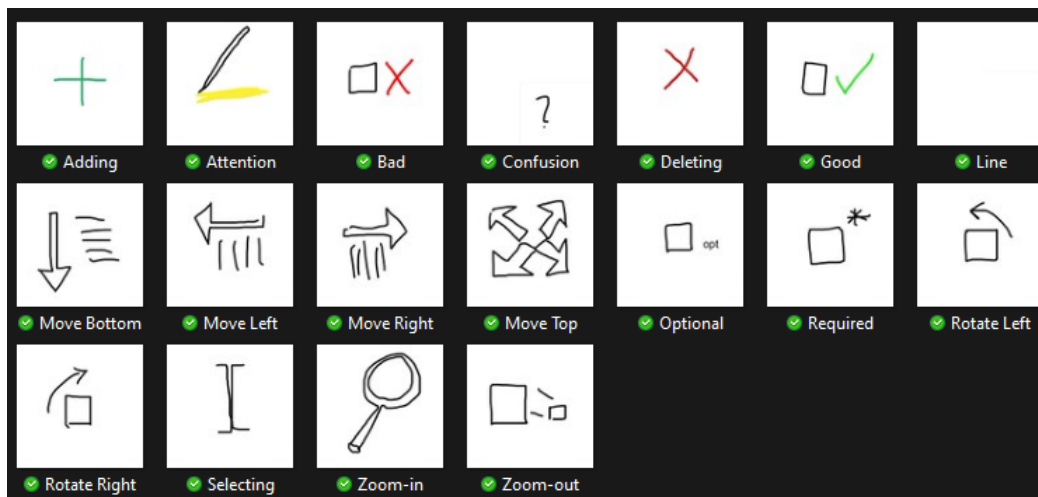


Figure B.12: Participant N°12

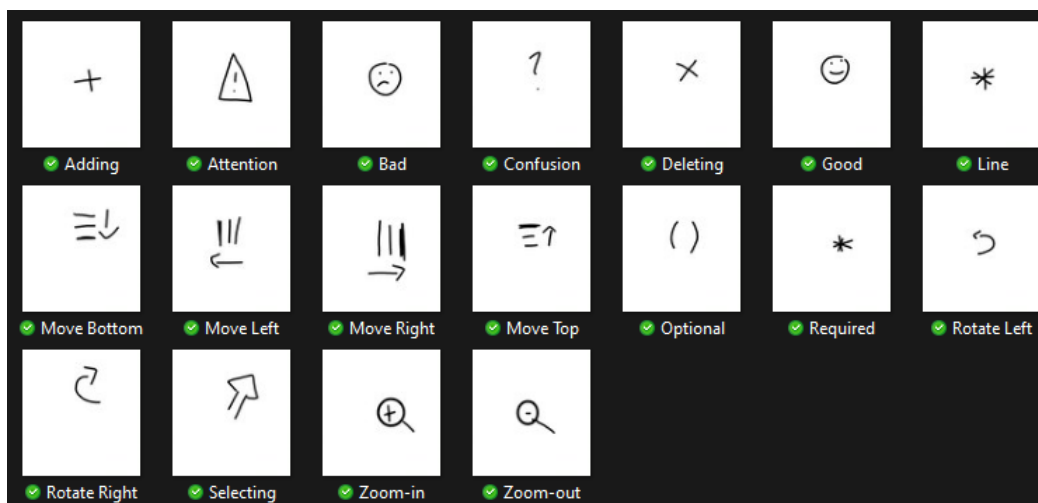


Figure B.13: Participant N°13

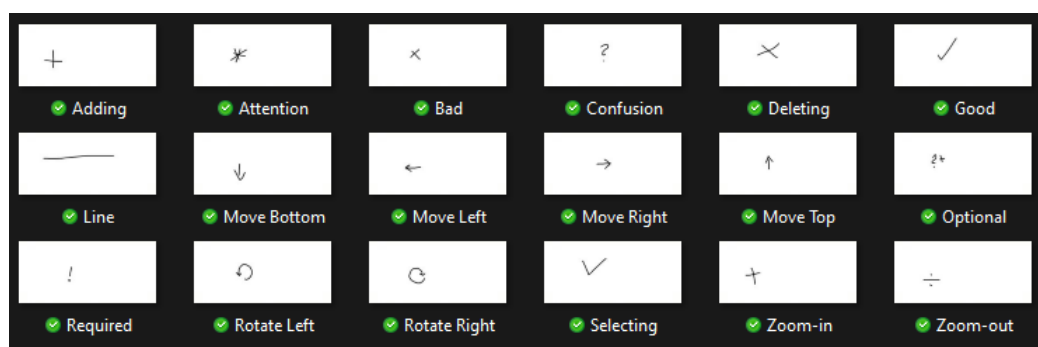


Figure B.14: Participant N°14

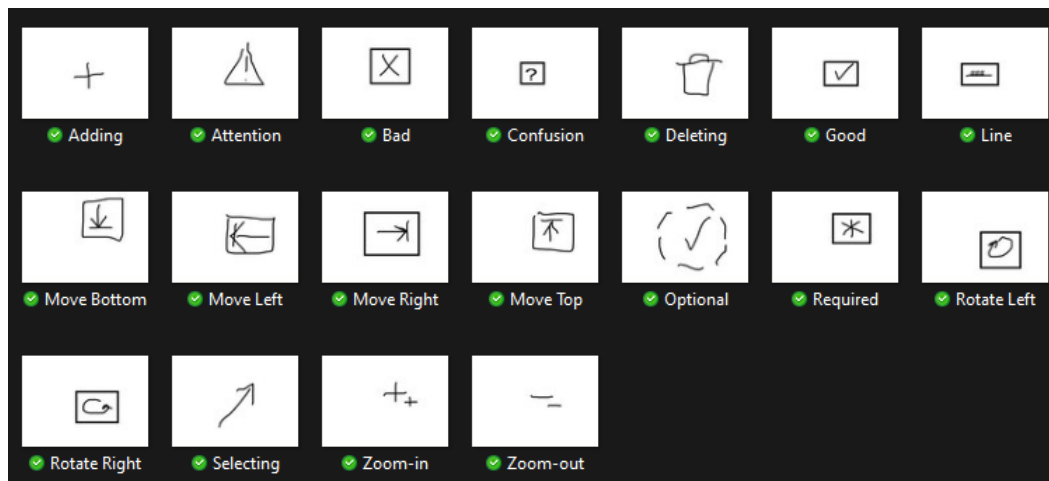


Figure B.15: Participant N°15

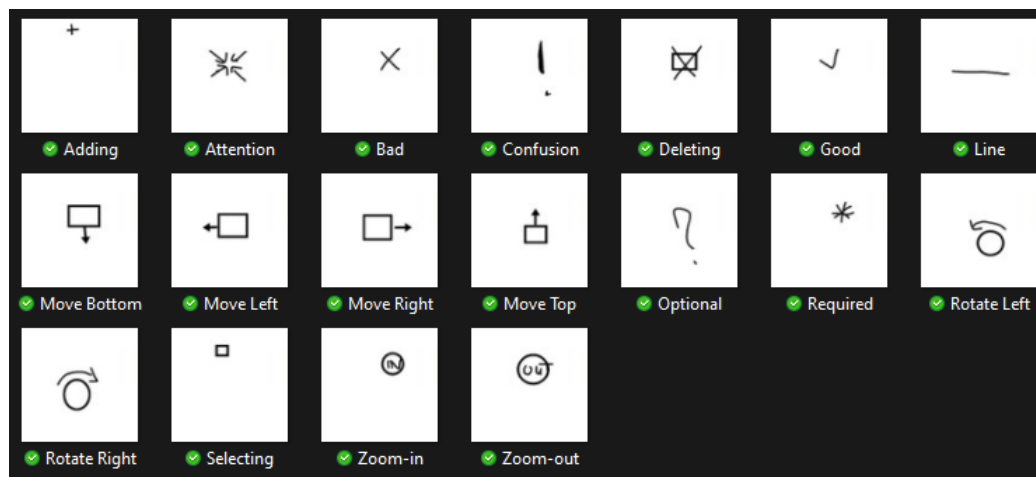


Figure B.16: Participant N°16

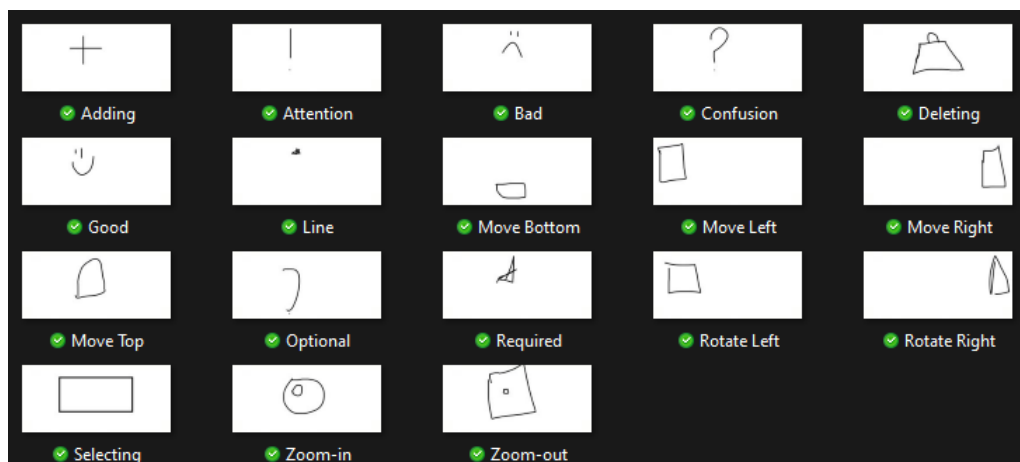


Figure B.17: Participant N°17

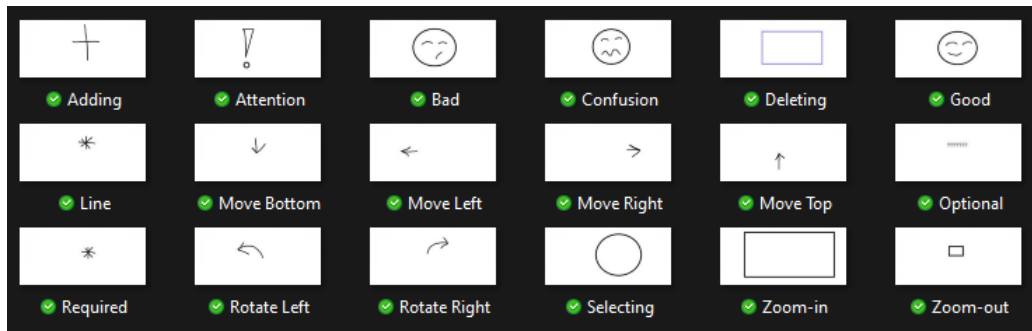


Figure B.18: Participant N°18

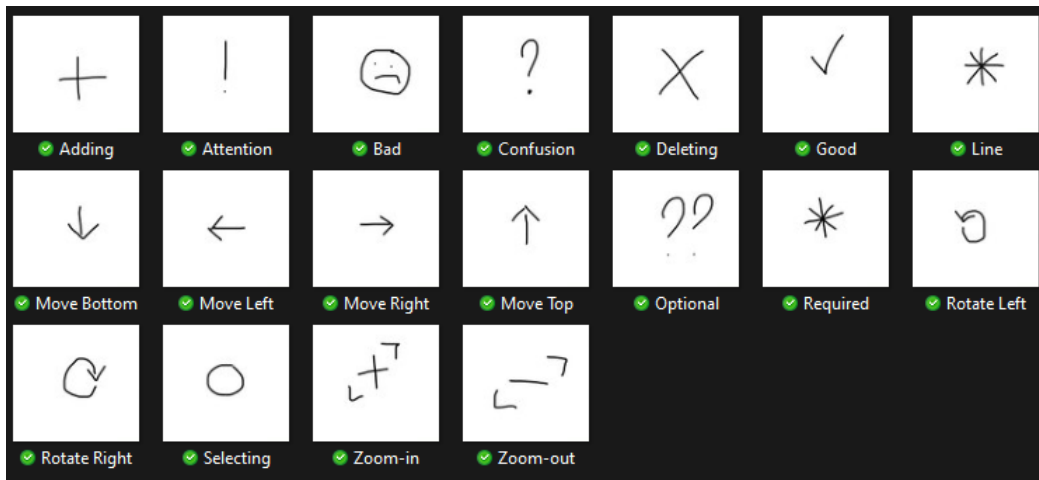


Figure B.19: Participant N°19

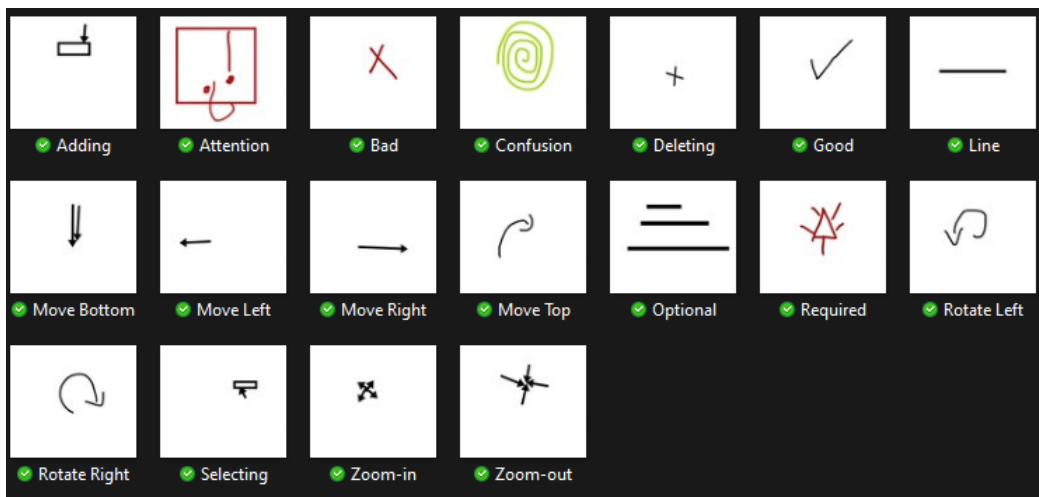


Figure B.20: Participant N°20

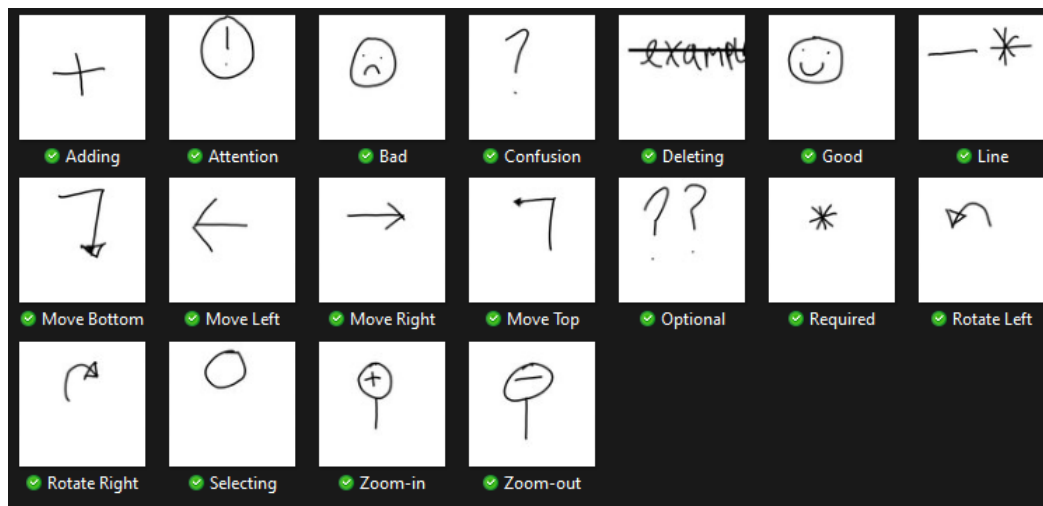


Figure B.21: Participant N°21

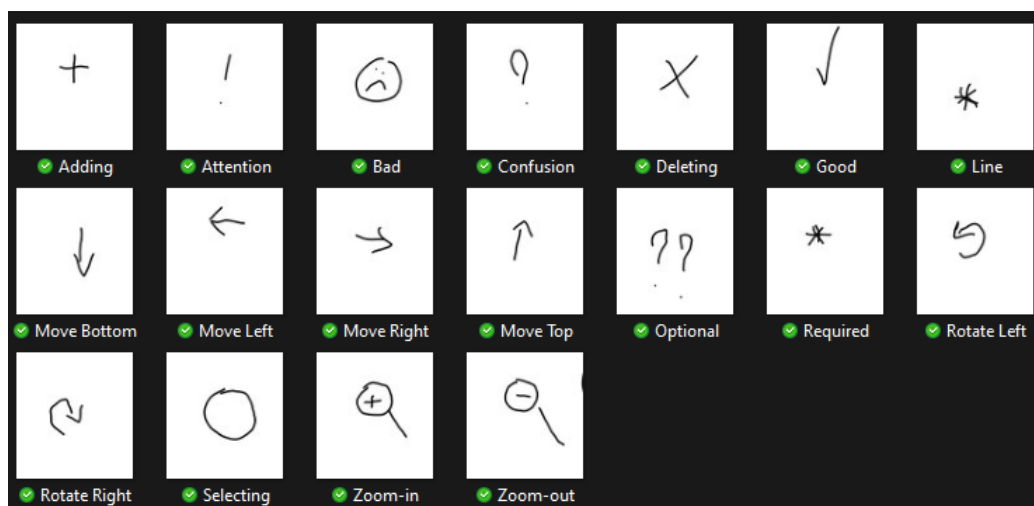


Figure B.22: Participant N°22

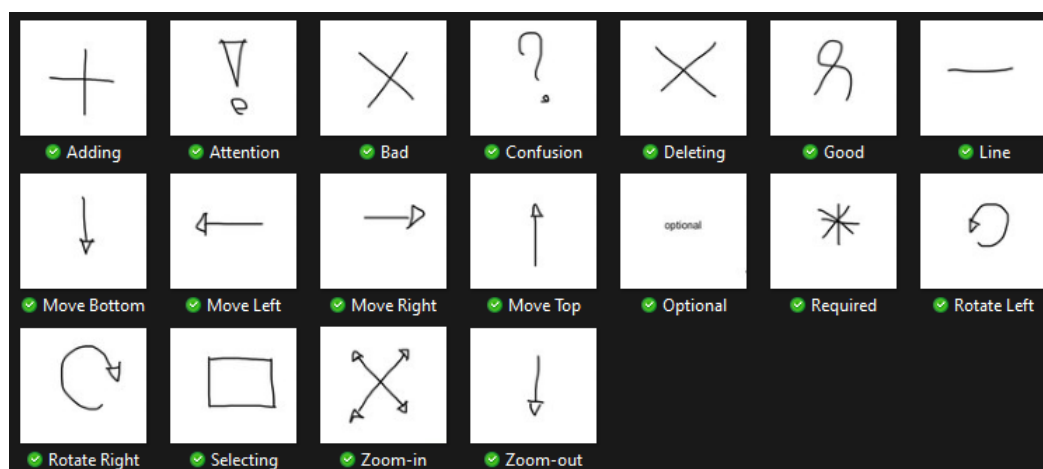


Figure B.23: Participant N°23

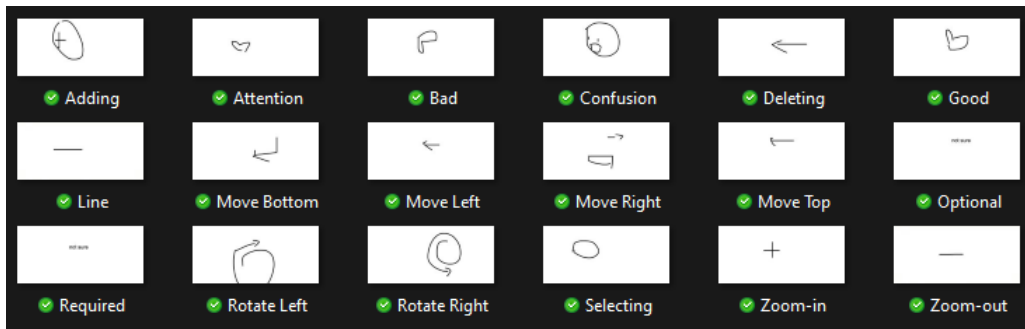


Figure B.24: Participant N°24

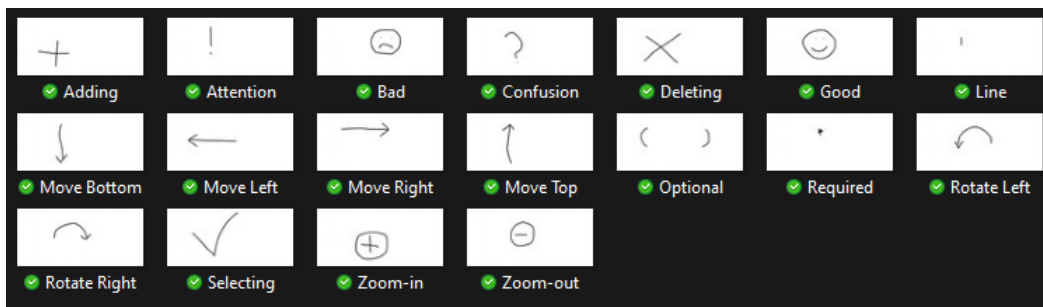


Figure B.25: Participant N°25

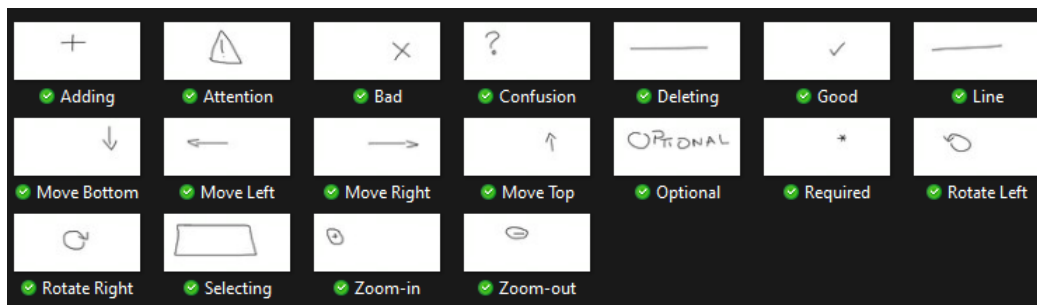


Figure B.26: Participant N°26

Appendix C

Classification Groups for each Annotation Type

This appendix regroups the classification groups created in Chapter 3

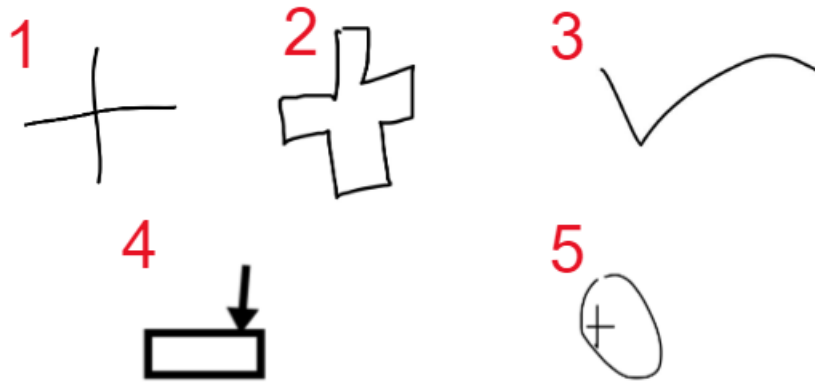


Figure C.1: Annotation Type: Add

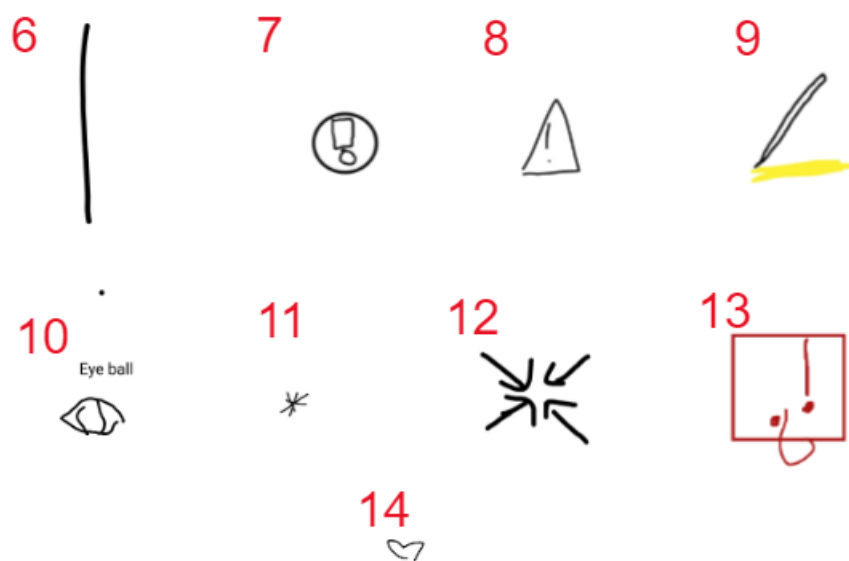


Figure C.2: Annotation Type: Attention

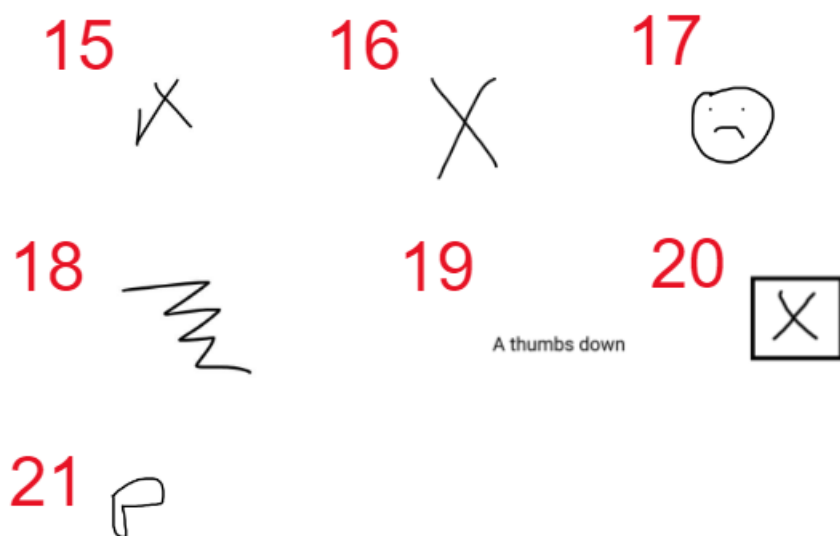


Figure C.3: Annotation Type: Bad



Figure C.4: Annotation Type: Confusion

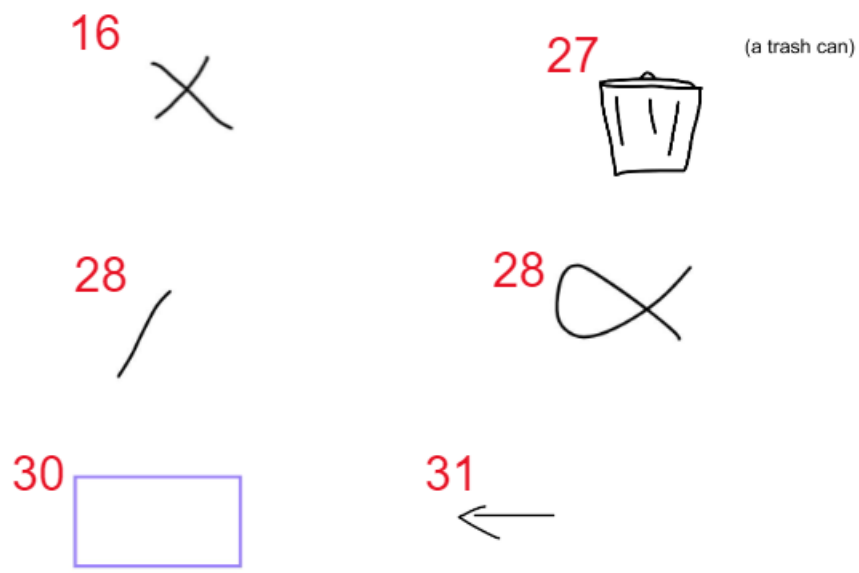


Figure C.5: Annotation Type: Delete

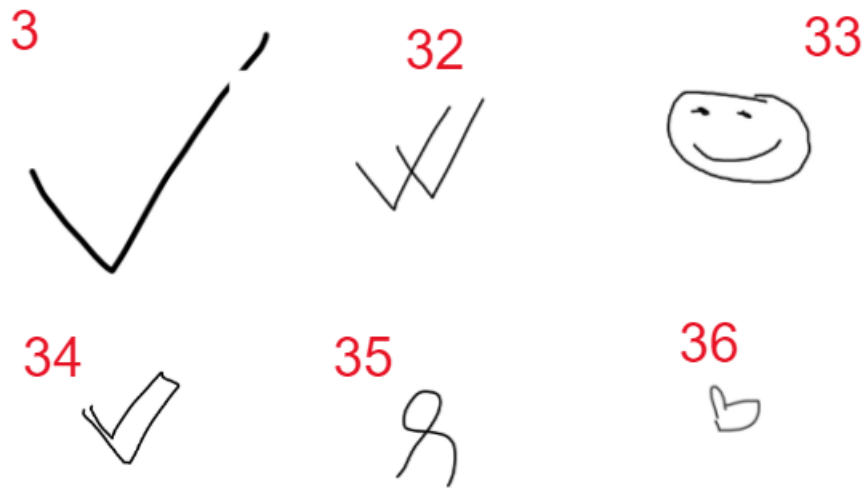


Figure C.6: Annotation Type: Good



Figure C.7: Annotation Type: Line



Figure C.8: Annotation Type: Move Bottom

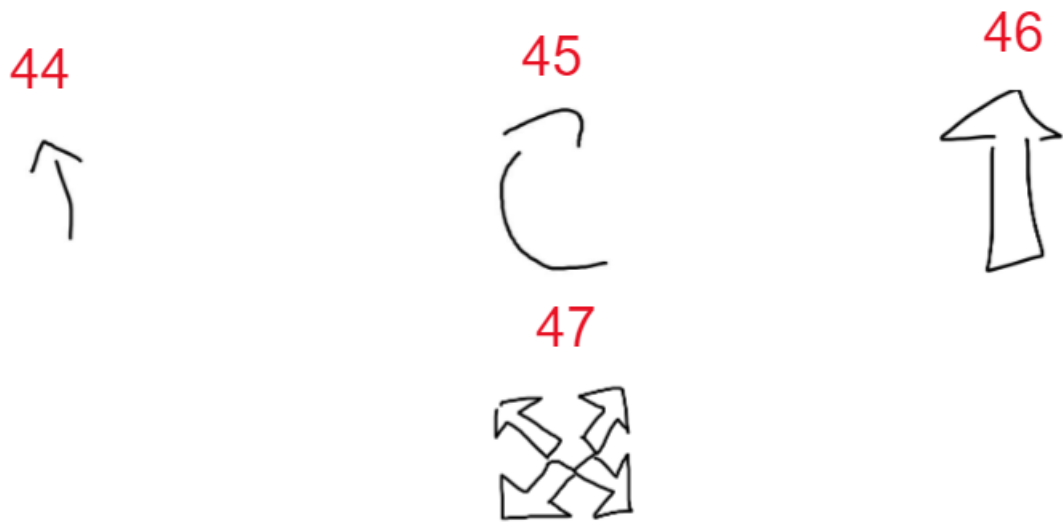


Figure C.9: Annotation Type: Move Top

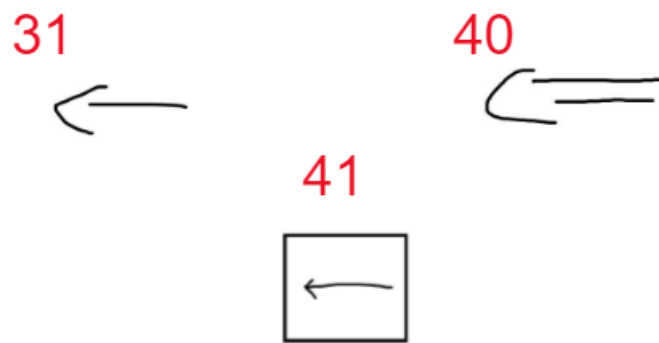


Figure C.10: Annotation Type: Move Left



Figure C.11: Annotation Type: Move Right

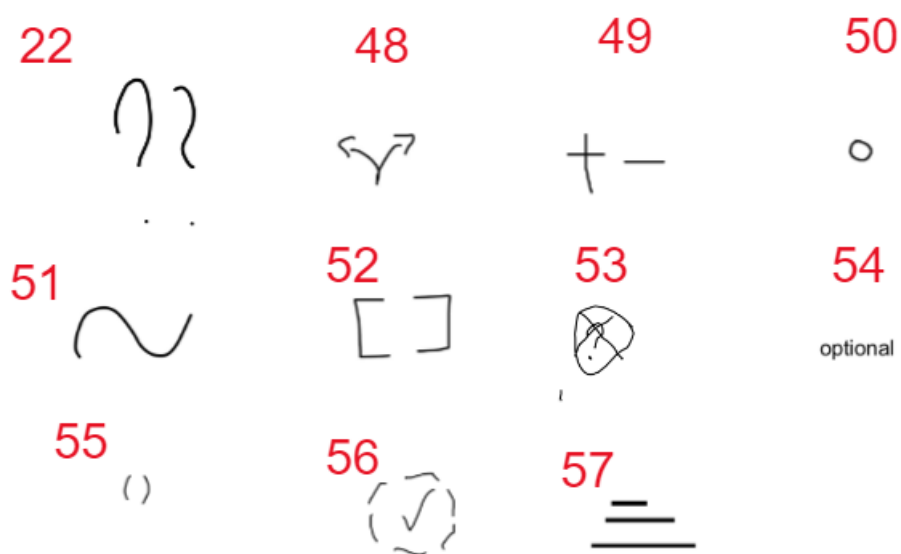


Figure C.12: Annotation Type: Optional



Figure C.13: Annotation Type: Required



Figure C.14: Annotation Type: Rotate Left

62

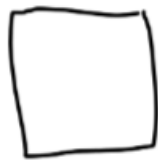


64



Figure C.15: Annotation Type: Rotate Right

30



52



65

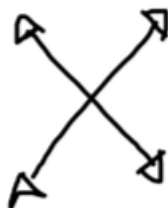


66



Figure C.16: Annotation Type: Select

47



67



68



69



Figure C.17: Annotation Type: Zoom-in



Figure C.18: Annotation Type: Zoom-in

Appendix D

Annotation Types Recognition on *Crowditivity*

This appendix shows all the annotation types defined in Chapter 4 being recognized on our platform *Crowditivity*.

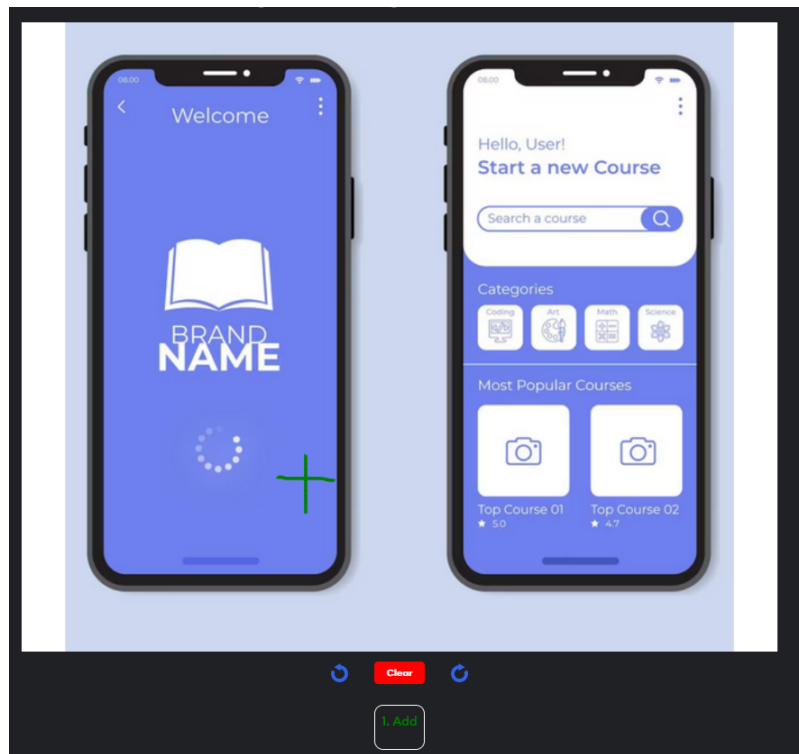


Figure D.1: Annotation Type Recognition: Add

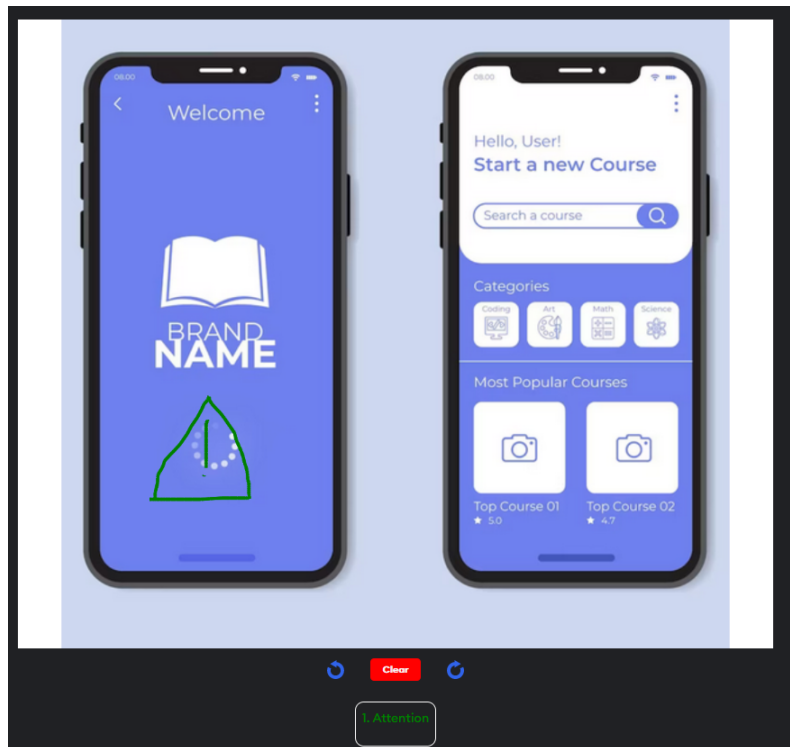


Figure D.2: Annotation Type Recognition: Attention

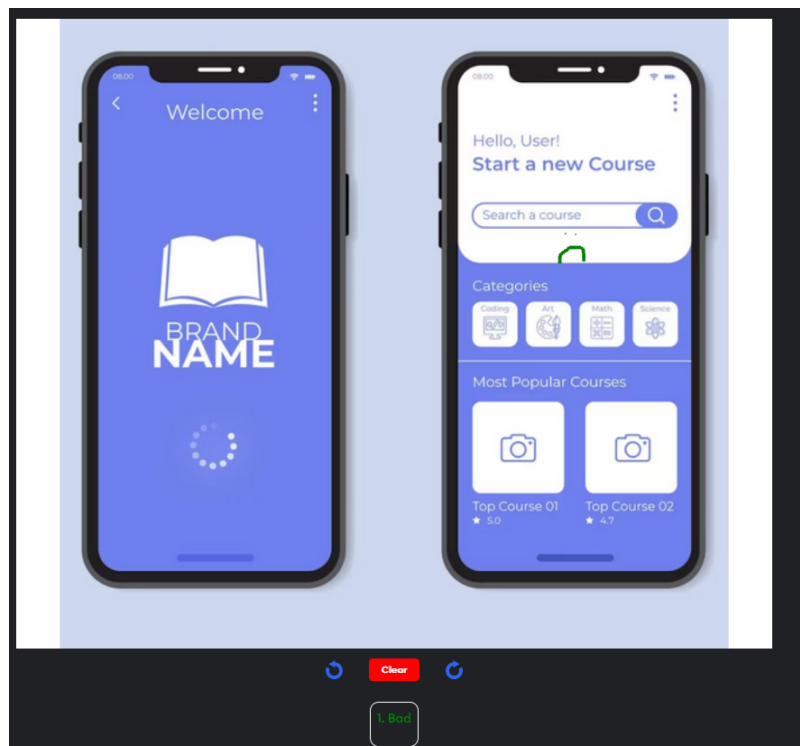


Figure D.3: Annotation Type Recognition: Bad

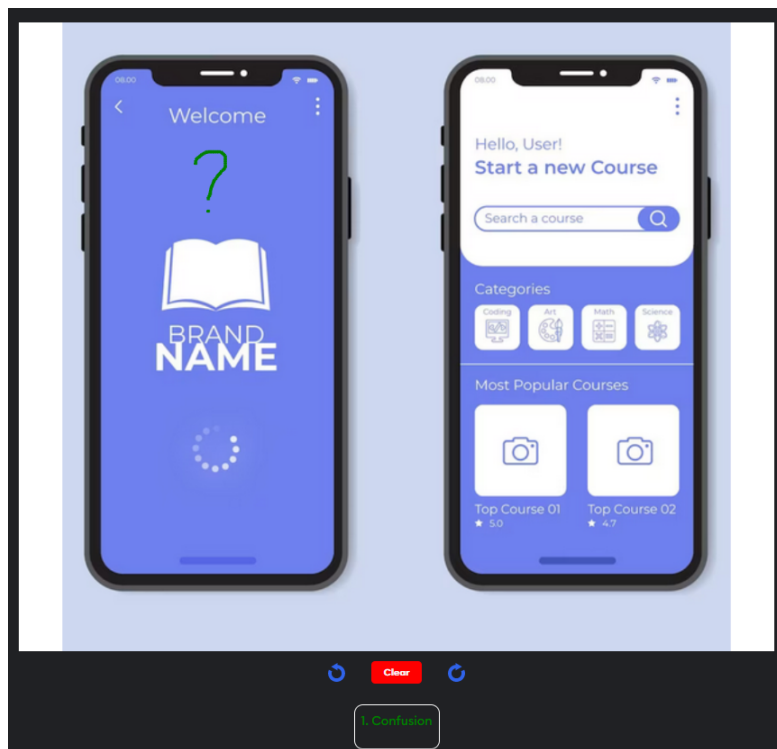


Figure D.4: Annotation Type Recognition: Confusion

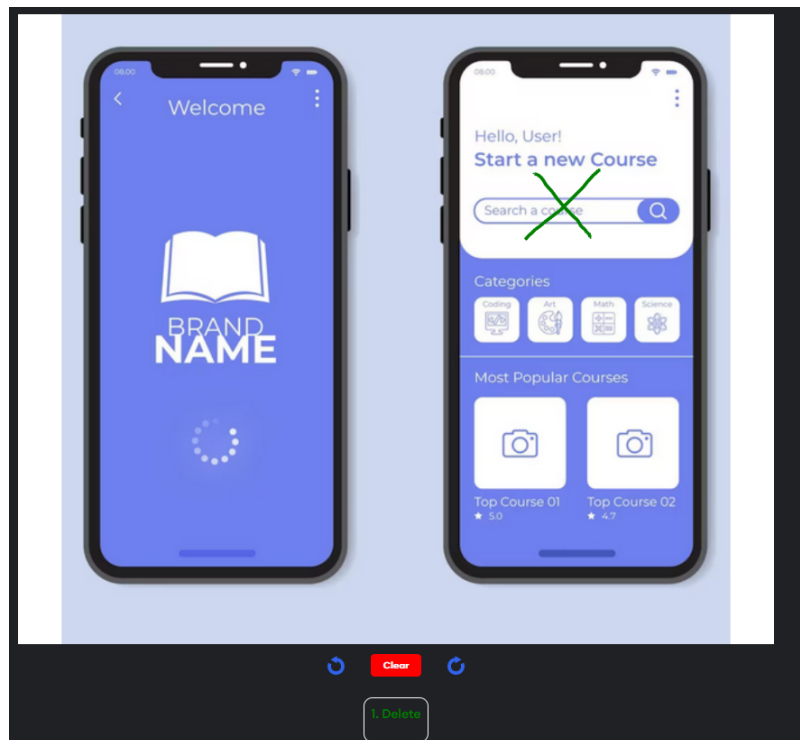


Figure D.5: Annotation Type Recognition: Delete

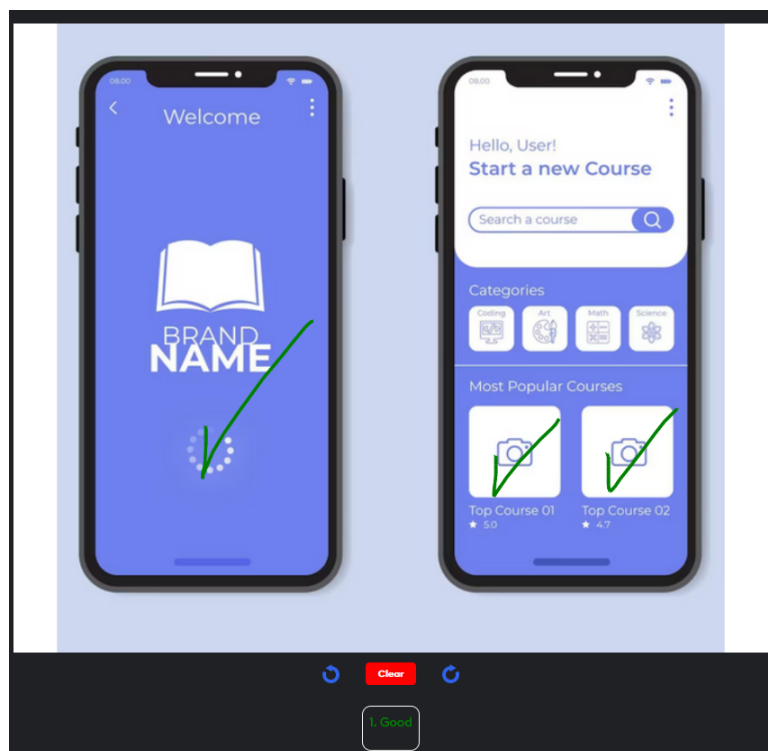


Figure D.6: Annotation Type Recognition: Good

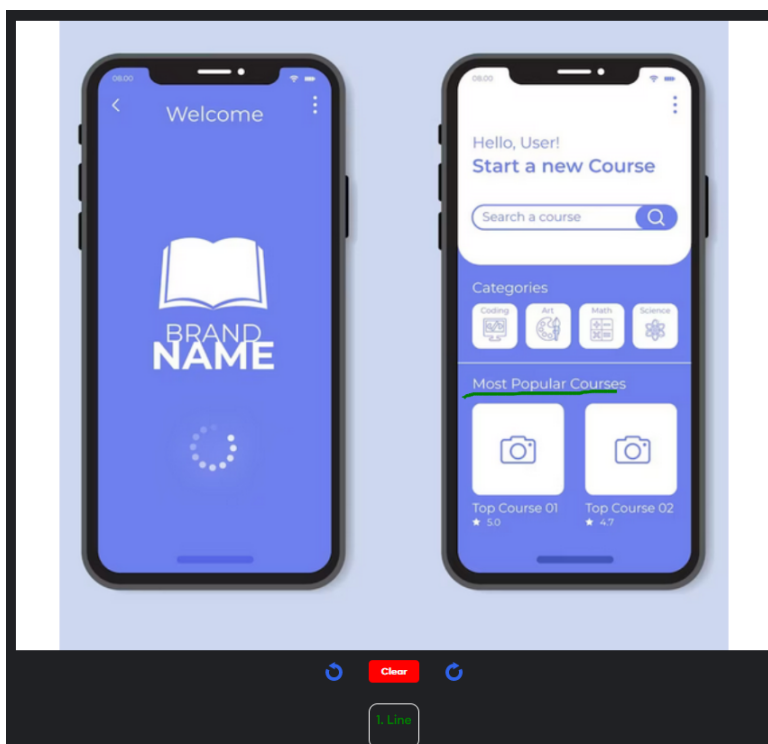


Figure D.7: Annotation Type Recognition: Line

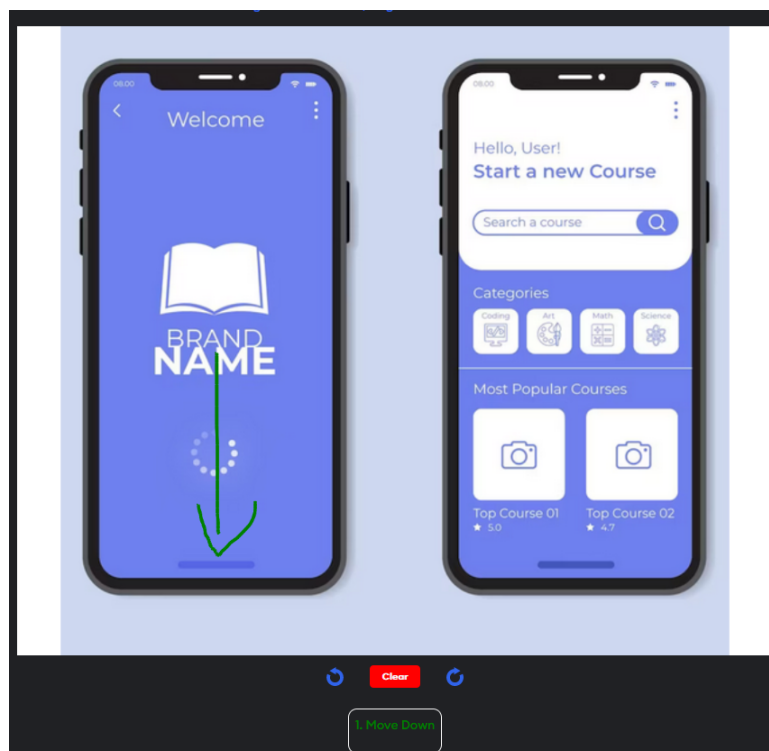


Figure D.8: Annotation Type Recognition: Move Bottom

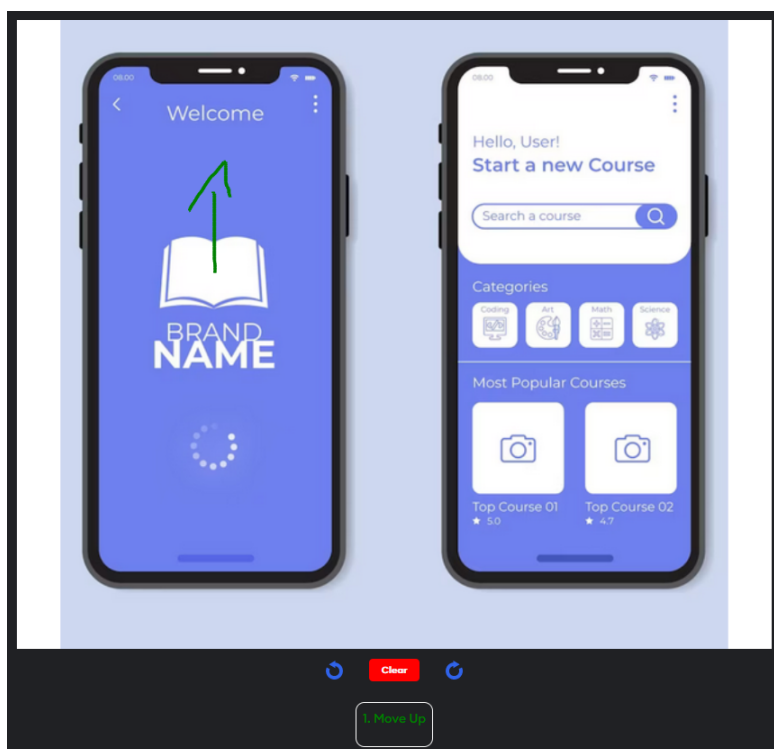


Figure D.9: Annotation Type Recognition: Move Top

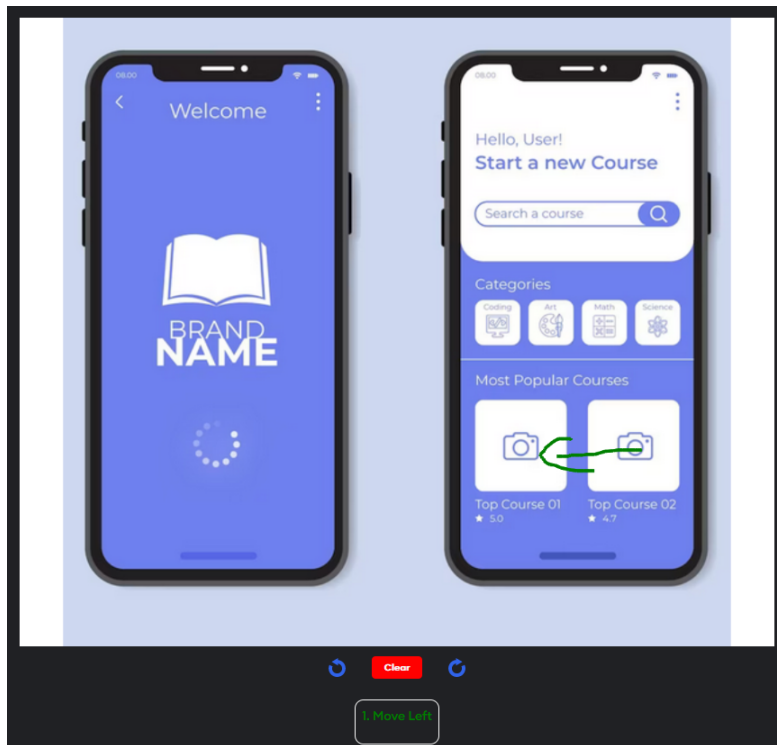


Figure D.10: Annotation Type Recognition: Move Left

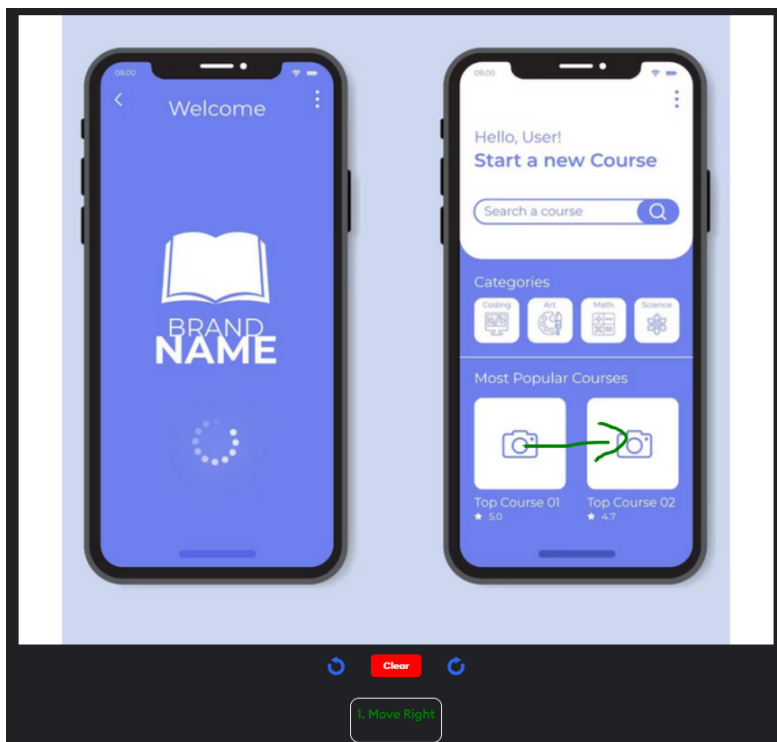


Figure D.11: Annotation Type Recognition: Move Right

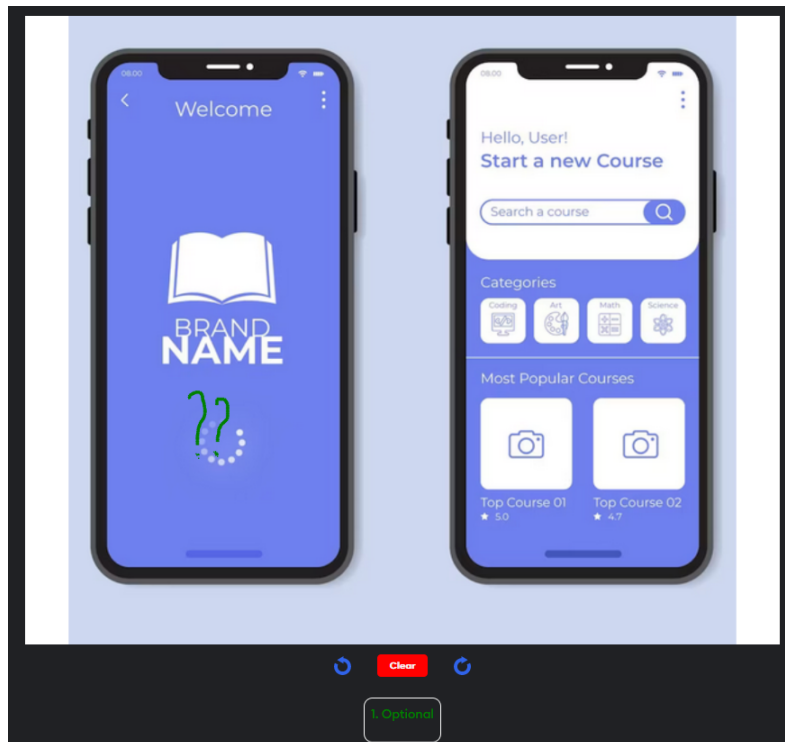


Figure D.12: Annotation Type Recognition: Optional

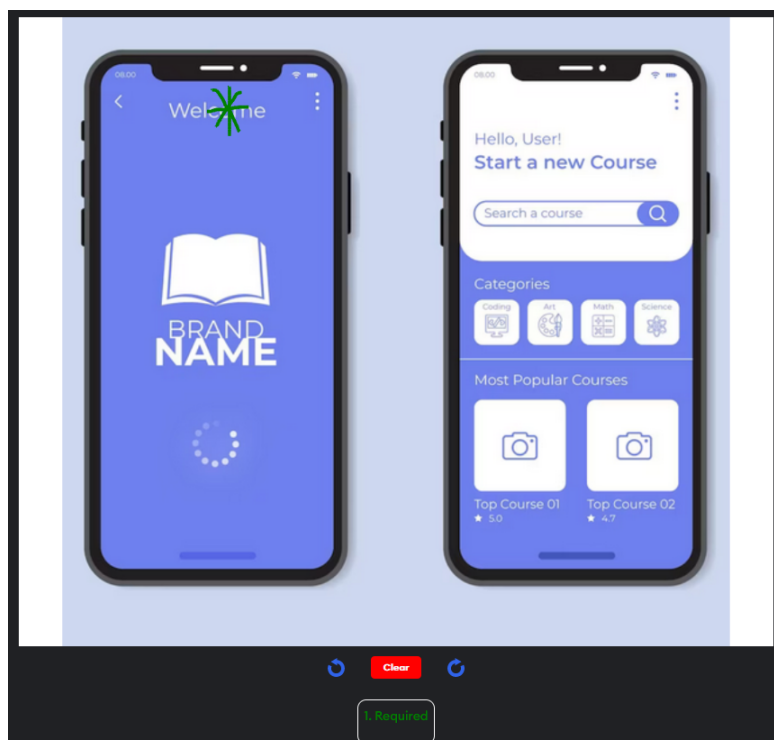


Figure D.13: Annotation Type Recognition: Required

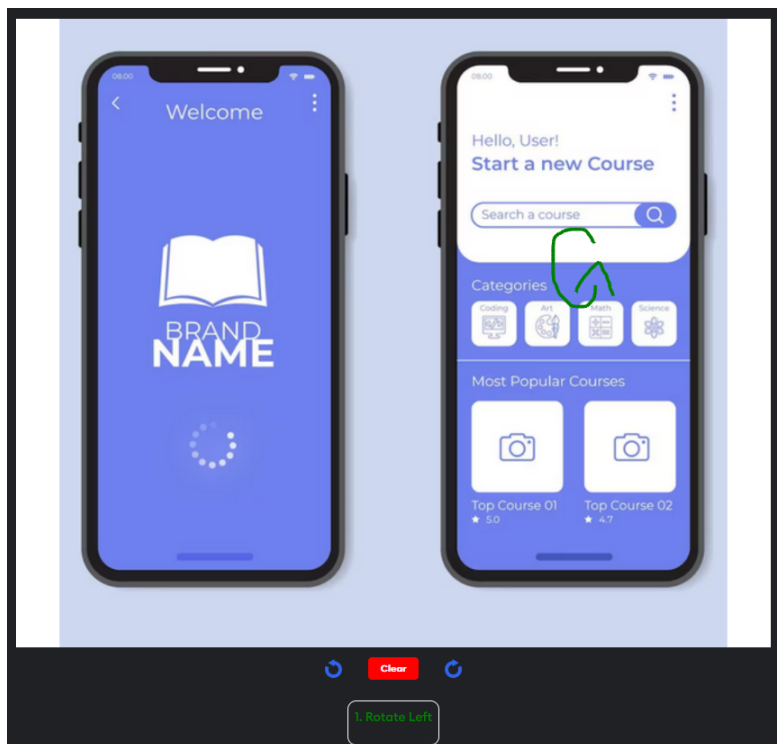


Figure D.14: Annotation Type Recognition: Rotate Left

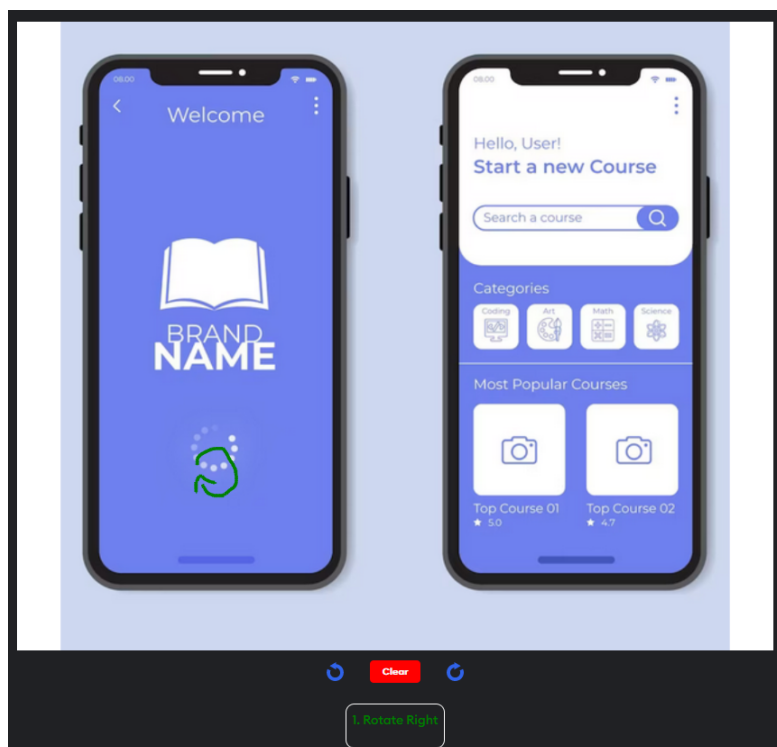


Figure D.15: Annotation Type Recognition: Rotate Right

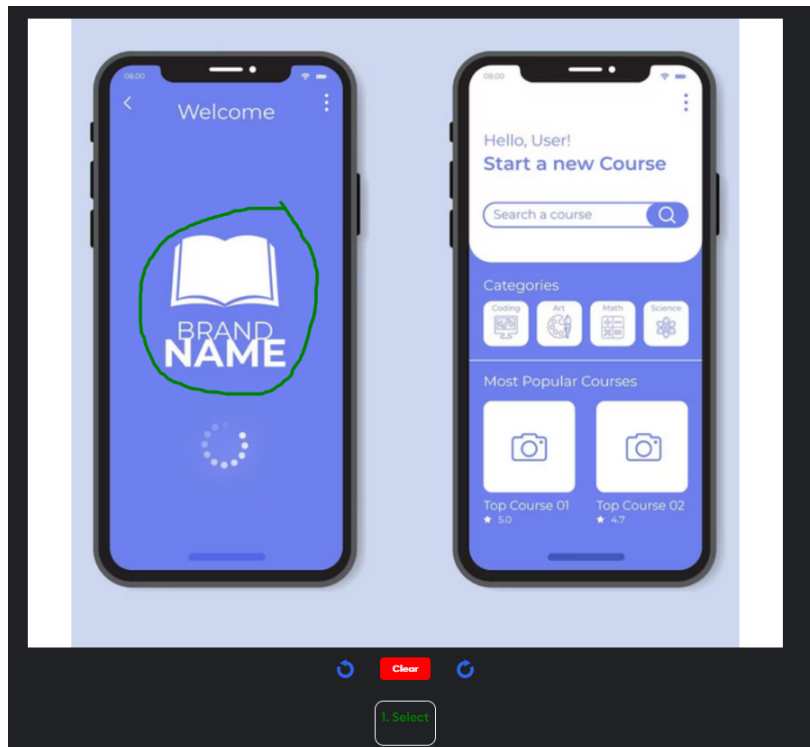


Figure D.16: Annotation Type Recognition: Select

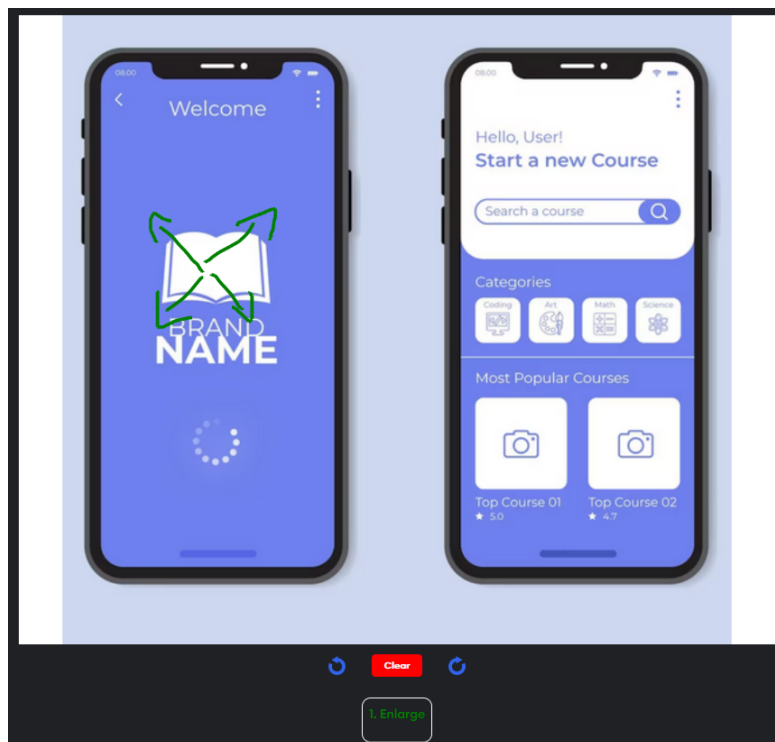


Figure D.17: Annotation Type Recognition: Zoom-in

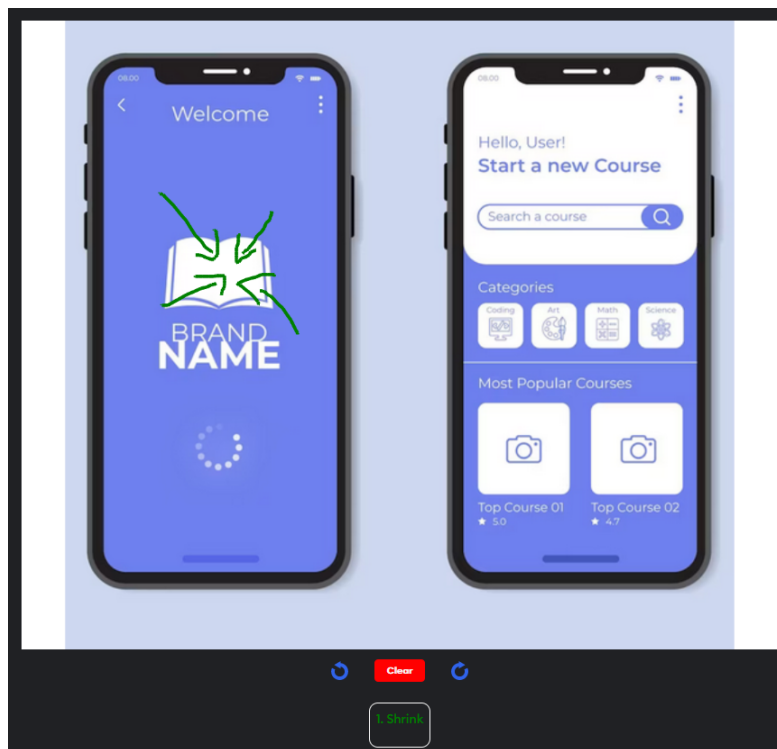


Figure D.18: Annotation Type Recognition: Zoom-in

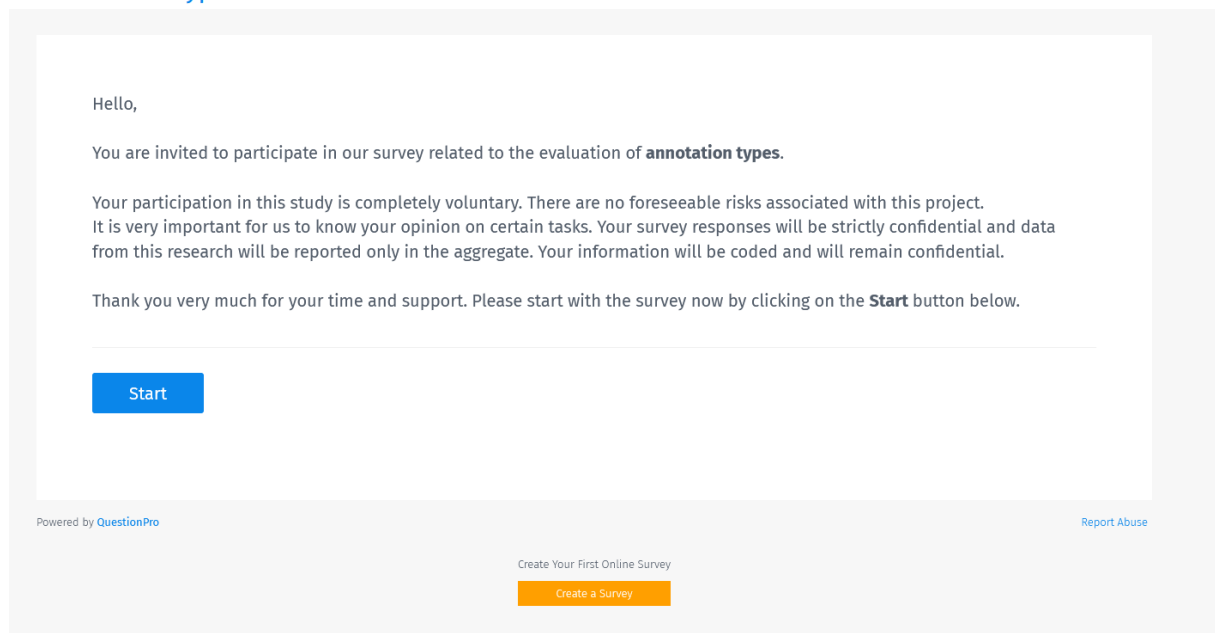
Appendix E

Examples of Utilized Forms and Surveys

This appendix regroups all the forms and surveys created to elicit, identify, and UEQ+ used in this thesis.

E.0.1 QuestionPro

Annotation Types



The screenshot shows a survey introduction page from QuestionPro. The page has a light gray background. At the top, there is a white box containing the following text: "Hello," followed by "You are invited to participate in our survey related to the evaluation of **annotation types**." Below this, it states: "Your participation in this study is completely voluntary. There are no foreseeable risks associated with this project. It is very important for us to know your opinion on certain tasks. Your survey responses will be strictly confidential and data from this research will be reported only in the aggregate. Your information will be coded and will remain confidential." Finally, it says: "Thank you very much for your time and support. Please start with the survey now by clicking on the **Start** button below." Below the text is a blue button labeled "Start". At the bottom of the page, there is a footer with the text "Powered by QuestionPro" on the left, "Create Your First Online Survey" in the center, and "Report Abuse" on the right. Below the center text is an orange button labeled "Create a Survey".

Figure E.1: Question Pro - Identification Survey: Introduction

• What is your gender?

-- Select --

• I use a computer frequently

Disagree Agree

1 2 3 4 5 6 7

☐ ☐ ☐ ☐ ☐ ☐ ☐

• I use a smartphone frequently

Disagree Agree

1 2 3 4 5 6 7

☐ ☐ ☐ ☐ ☐ ☐ ☐

I use a tablet frequently

Disagree Agree

1 2 3 4 5 6 7

☐ ☐ ☐ ☐ ☐ ☐ ☐

In the following questionnaire, several images of annotation types will be displayed. Your task is to select the element that is the most appropriate according to the instruction describing the types to be used to perform such action.

[<](#) [Next](#)

Figure E.2: Question Pro - Identification Survey: Demographic Information

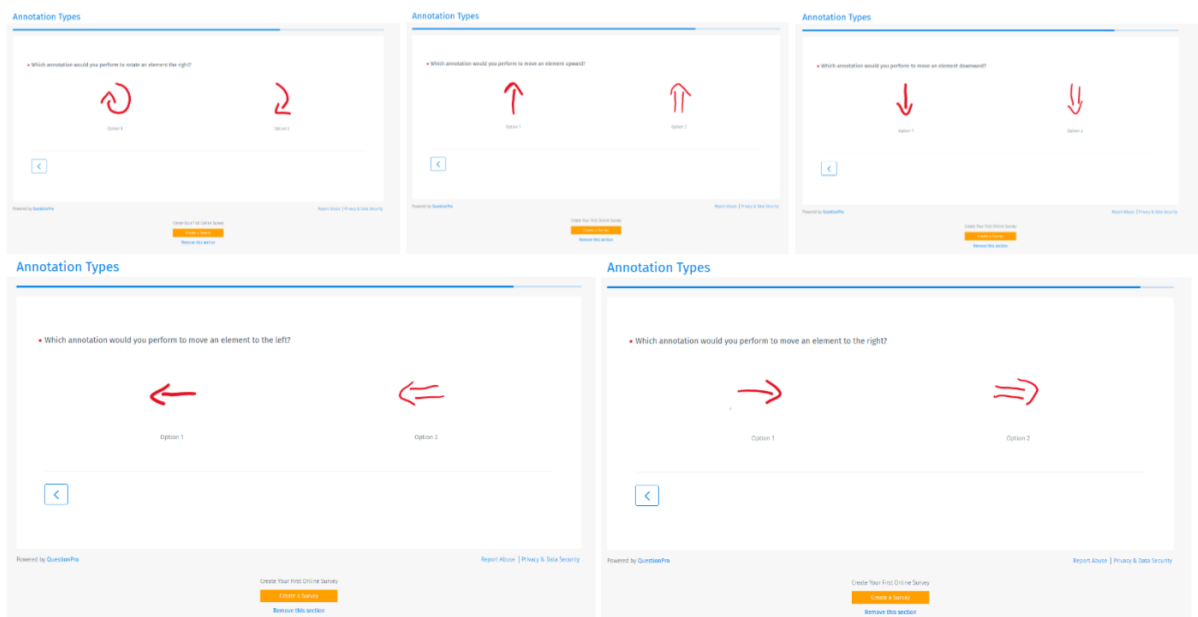


Figure E.3: Question Pro - Identification Survey: Questions I

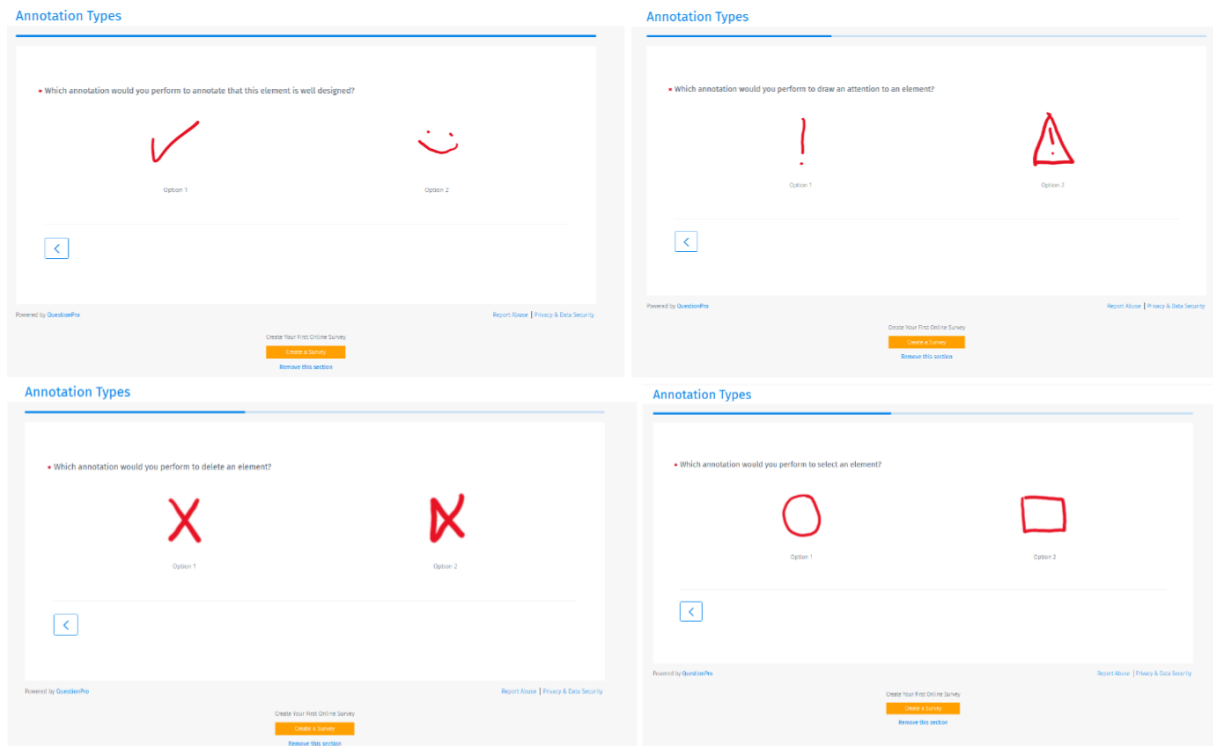


Figure E.4: Question Pro - Identification Survey: Questions II

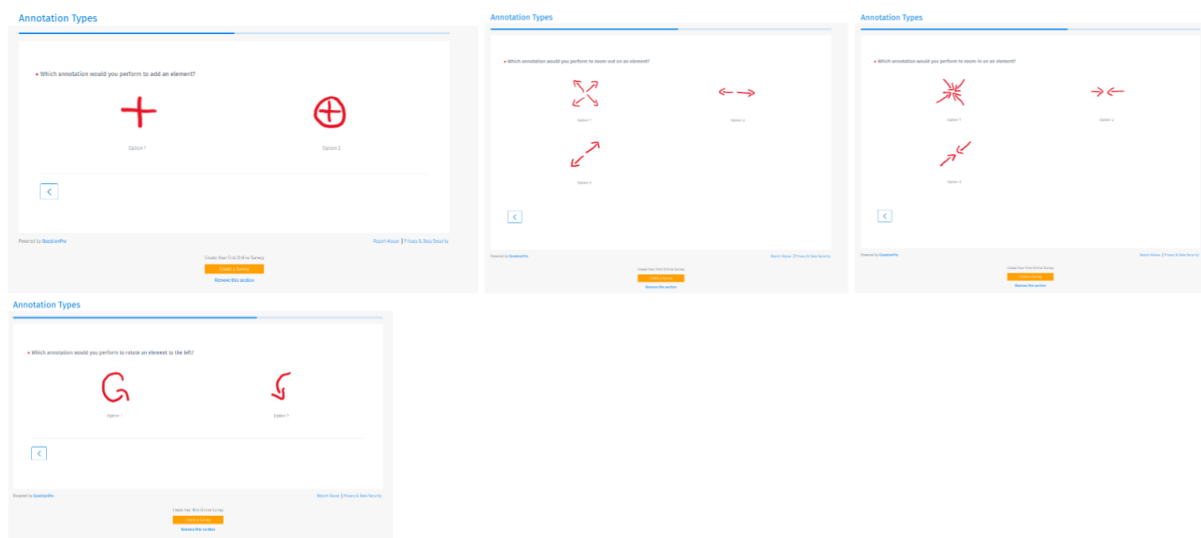


Figure E.5: Question Pro - Identification Survey: Questions III

E.0.2 Forms.app

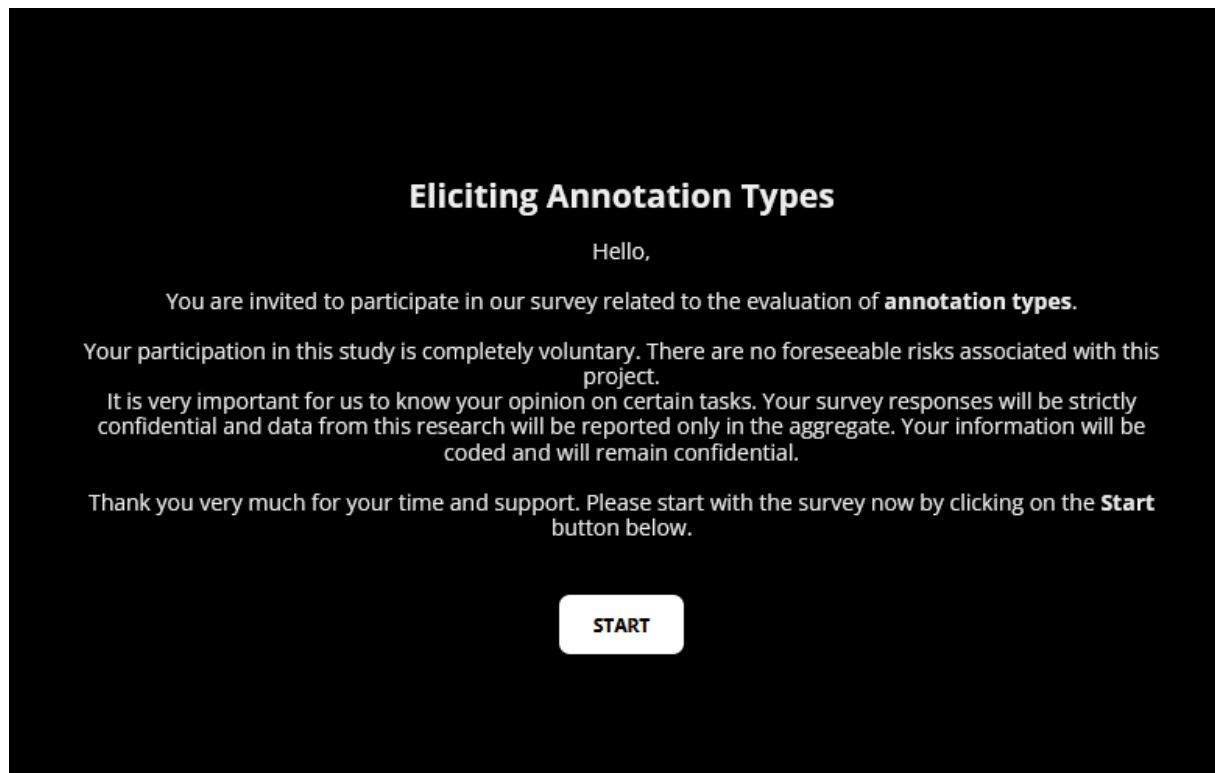


Figure E.6: Forms.app - Elicitation Study: Introduction

Elicitation of Annotation Types

1. What is your age? *

↑?

18+

2. What is your gender? *

Please select

3. I use a computer frequently *

1

2

3

4

5

6

7

4. I use a smartphone frequently *

1

2

3

4

5

6

7

5. I use a tablet frequently *

1

2

3

4

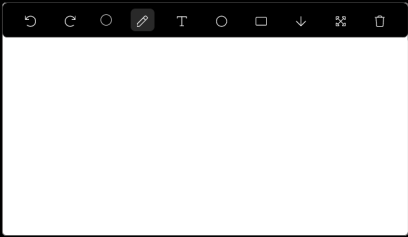
5

6

7

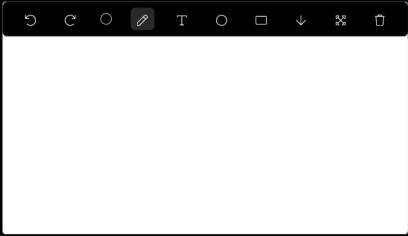
Figure E.7: Forms.app - Elicitation Study: Demographic Information

for example: circle, square, triangle...



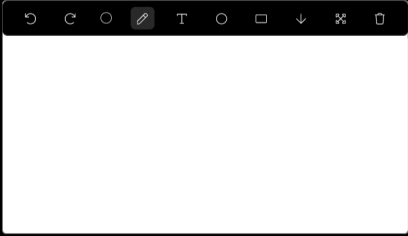
7. Please draw a symbol for deleting an element: *

for example: cross, lines...



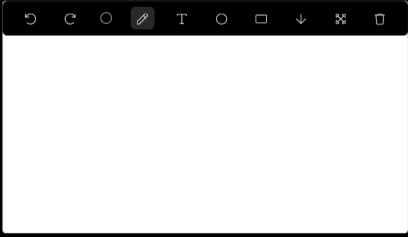
8. Please draw a symbol for adding an element: *

for example: +, plus...

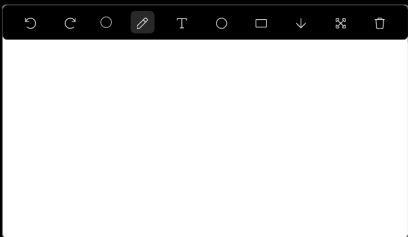


9. Please draw a symbol for showing attention to an element: *

for example: !, exclamation mark...



10. Please draw a symbol to enlarge/zoomIn on an element: *



11. Please draw a symbol to shrink/zoomOut on an element: *

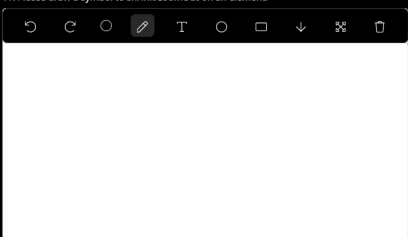


Figure E.8: Forms.app - Elicitation Study: Questions I

12. Please draw a symbol to move an element to the TOP: *

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

13. Please draw a symbol to move an element to the BOTTOM: *

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

14. Please draw a symbol to move an element to the LEFT: *

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

15. Please draw a symbol to move an element to the RIGHT: *

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

16. Please draw a symbol to show an element is good: *
check mark, happy face...

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

17. Please draw a symbol to show an element is bad: *
sad face...

↶ ↷ ○  T ○ □ ↓ ⌘ 🗑️

Figure E.9: Forms.app - Elicitation Study: Questions II

18. Please draw a symbol to rotate an element LEFT: *

↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

19. Please draw a symbol to rotate an element RIGHT: *

↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

20. Please draw a symbol to show that this element is required: *

Example: asterisk
↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

21. Please draw a line: *

↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

22. Please draw to show confusion on an element: *

Example: ?
↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

23. Please draw to show that this element can be OPTIONAL: *

Example: ?
↶ ↷ ○ ✎ T ○ □ ↓ ⌘ 🗑

Submit

Figure E.10: Forms.app - Elicitation Study: Questions III

E.0.3 Google Forms

Crowditivity - User Experience Questionnaire (UEQ+)

Hello!

Your valuable input plays a crucial role in enhancing the platform's usability and improving the overall research experience. Please complete the following questionnaire to assess the product. The questionnaire contains opposing pairs of product properties. The grades between the opposites are indicated by circles. Check one of the circles to indicate your level of agreement with the individual terms.

Thank you for being part of our questionnaire!

Figure E.11: Google Forms - UEQ+: Introduction

Q1: What is you age? *

Your answer

Q2: What is your gender? *

Choose

Q3: I use a smartphone frequently on a scale from 1 to 7? *

1 2 3 4 5 6 7

Strongly disagree Strongly agree

Q4: I use a computer frequently on a scale from 1 to 7? *

1 2 3 4 5 6 7

Strongly disagree Strongly agree

Q5: I use a tablet frequently on a scale from 1 to 7? *

1 2 3 4 5 6 7

Strongly disagree Strongly agree

Figure E.12: Google Forms - UEQ+: Demographic Information

Q6.a: In my opinion, the product is generally *

	1	2	3	4	5	6	7	
bad	☐	☐	☐	☐	☐	☐	☐	good

Q6.b: In my opinion, the product is generally *

	1	2	3	4	5	6	7	
unfriendly	☐	☐	☐	☐	☐	☐	☐	friendly

Q7.a: To achieve my goals, I consider the product as *

	1	2	3	4	5	6	7	
slow	☐	☐	☐	☐	☐	☐	☐	fast

Q7.b: To achieve my goals, I consider the product as *

	1	2	3	4	5	6	7	
impractical	☐	☐	☐	☐	☐	☐	☐	practical

Q8.a: In my opinion, handling and working with the product are *

	1	2	3	4	5	6	7	
boring	☐	☐	☐	☐	☐	☐	☐	exiting

Q8.b: In my opinion, handling and working with the product are *

	1	2	3	4	5	6	7	
not interesting	☐	☐	☐	☐	☐	☐	☐	interesting

Figure E.13: Google Forms - UEQ+: Questions I

Q9.a: In my opinion, the idea behind the product and its design are *

	1	2	3	4	5	6	7	
dull	☐	☐	☐	☐	☐	☐	☐	creative

Q9.b: In my opinion, the idea behind the product and its design are *

	1	2	3	4	5	6	7	
conservative	☐	☐	☐	☐	☐	☐	☐	innovative

Q10.a: I consider the possibility of using the product as *

	1	2	3	4	5	6	7	
useless	☐	☐	☐	☐	☐	☐	☐	useful

Q10.b: I consider the possibility of using the product as *

	1	2	3	4	5	6	7	
not beneficial	☐	☐	☐	☐	☐	☐	☐	beneficial

Q11.a: I generally consider the design of the product as *

	1	2	3	4	5	6	7	
not elegant	☐	☐	☐	☐	☐	☐	☐	elegant

Q11.b: I generally consider the design of the product as *

	1	2	3	4	5	6	7	
not presentable	☐	☐	☐	☐	☐	☐	☐	presentable

Figure E.14: Google Forms - UEQ+: Questions II

Q12.a: In my opinion, the visual design of the product is *

1

2

3

4

5

6

7

ugly

☐

☐

☐

☐

☐

☐

☐

beautiful

Q12.b: In my opinion, the visual design of the product is *

1

2

3

4

5

6

7

unappealing

☐

☐

☐

☐

☐

☐

☐

appealing

Q13.a: In my opinion, using the product is *

1

2

3

4

5

6

7

difficult

☐

☐

☐

☐

☐

☐

☐

easy

Q13.b: In my opinion, using the product is *

1

2

3

4

5

6

7

illogical

☐

☐

☐

☐

☐

☐

☐

logical

Q14.a: In my opinion the user interface of the product looks *

1

2

3

4

5

6

7

unstructured

☐

☐

☐

☐

☐

☐

☐

structured

Q14.b: In my opinion the user interface of the product looks *

1

2

3

4

5

6

7

poorly grouped

☐

☐

☐

☐

☐

☐

☐

well grouped

Submit

Page 1 of 1

Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms

Figure E.15: Google Forms - UEQ+: Questions III

Appendix F

GDPR Consent

This appendix is for the documentation for GDPR Data Protection Agreement related to the data that were collected from the surveys conducted in this thesis.

Data Protection Agreement

(GDPR and EU Standard Contractual Clauses)

This Data Processing Agreement (“DPA”) is effective as of (“**May 28 2023**”) by and between **Crowditivity**, a **Belgium** corporation, (“**Company**”) acting on its own behalf and as agent for each Company Affiliate, and **QuestionPro Inc.**, a **California** Corporation, (“**Vendor**”) acting on its own behalf and as agent for each Vendor Affiliate, including but not limited to QuestionPro GMBH - Vendors’ subsidiary company in Germany.

This DPA supplements any terms of use or agreement (“**Principal Contracts**”) under which Vendor provides the Services.

In consideration of the mutual obligations set out herein, the parties hereby agree to be bound by the terms and conditions set out below.

1. Definitions

1.1 In this Addendum, the following terms shall have the meanings set out below and cognate terms shall be construed accordingly:

1.1.1 “**Applicable Laws**” means (a) European Union or Member State laws with respect to any Company Personal Data in respect of which any Company Group Member is subject to EU Data Protection Laws; and (b) any other applicable law with respect to any Company Personal Data in respect of which any Company Group Member is subject to any other Data Protection Laws;

1.1.2 “**Company Affiliate**” means an entity that owns or controls, is owned or controlled by or is or under common control or ownership with Company, where control is defined as the possession, directly or indirectly, of the power to direct or cause the direction of the management and policies of an entity, whether through ownership of voting securities, by contract or otherwise;

1.1.3 “**Company Group Member**” means Company or any Company Affiliate;

1.1.4 “**Company Personal Data**” means any Personal Data Processed by a Contracted Processor on behalf of a Company Group Member in connection with the Services;

1.1.5 “**Contracted Processor**” means Vendor or a Subprocessor;

1.1.6 “**Data Protection Laws and Regulations**” means all laws and regulations, including laws and regulations of the European Union, the European Economic Area and their member states, Switzerland, the United Kingdom and the United States and its states, applicable to the Processing of Personal Data under the Agreement.

1.1.7 “**EEA**” means the European Economic Area;

1.1.8 “**GDPR**” means EU General Data Protection Regulation 2016/679;

1.1.9 “**Restricted Transfer**” means:

1.1.9.1 a transfer of Company Personal Data from any Company Group Member to a Contracted Processor; or

1.1.9.2 an onward transfer of Company Personal Data from a Contracted Processor to a Contracted Processor, or between two establishments of a Contracted Processor,

in each case, where such transfer would be prohibited by Data Protection Laws (or by the terms of data transfer agreements put in place to address the data transfer restrictions of Data Protection Laws) in the absence of the Standard Contractual Clauses to be established under section 6.4.3 or 12 below;

1.1.10 “**Services**” means the services and other activities to be supplied to or carried out by or on behalf of Vendor for Company Group Members, including, data storage, co-location, internet access, hardware support and data backup services;

1.1.11 **"Standard Contractual Clauses"** means the contractual clauses set out in Annex 2, amended as indicated (in square brackets and italics) in that Annex and under section 13.4

1.1.12 **"Subprocessor"** means any person (including any third party and any Vendor Affiliate, but excluding an employee of Vendor or any of its sub-contractors) appointed by or on behalf of Vendor or any Vendor Affiliate to Process Personal Data on behalf of any Company Group Member in connection with the Services; and

1.1.13 **"Vendor Affiliate"** means an entity that owns or controls, is owned or controlled by or is or under common control or ownership with Vendor, where control is defined as the possession, directly or indirectly, of the power to direct or cause the direction of the management and policies of an entity, whether through ownership of voting securities, by contract or otherwise.

1.2 The terms, **"Commission," "Controller," "Data Subject," "Member State," "Personal Data," "Personal Data Breach," "Processing," "Processor"** and **"Supervisory Authority"** shall have the same meaning as in the GDPR, and their cognate terms shall be construed accordingly.

1.3 The word **"include"** shall be construed to mean include without limitation, and cognate terms shall be construed accordingly.

2. Authority

Vendor warrants and represents that, before any Vendor Affiliate Processes any Company Personal Data on behalf of any Company Group Member, Vendor's entry into this Addendum as agent for and on behalf of that Vendor Affiliate will have been duly and effectively authorised (or subsequently ratified) by that Vendor Affiliate.

3. Processing of Company Personal Data

3.1 Vendor shall act as Processor to the Company who may act either as Controller or Processor with respect to Company Personnel Data.

3.2 Vendor and each Vendor Affiliate shall:

3.2.1 comply with all applicable Data Protection Laws in the Processing of Company Personal Data; and

3.2.2 not Process Company Personal Data other than on the relevant Company Group Member's documented instructions unless Processing is required by Applicable Laws to which the relevant Contracted Processor is subject, in which case Vendor or the relevant Vendor Affiliate shall to the extent permitted by Applicable Laws inform the relevant Company Group Member of that legal requirement before the relevant Processing of that Personal Data.

3.3 Each Company Group Member:

3.3.1 instructs Vendor and each Vendor Affiliate (and authorises Vendor and each Vendor Affiliate to instruct each Subprocessor) to:

3.3.1.1 Process Company Personal Data; and

3.3.1.2 in particular, transfer Company Personal Data to any country or territory,

as reasonably necessary for the provision of the Services and consistent with the Principal Contracts; and

3.3.2 warrants and represents that it is and will at all relevant times remain duly and effectively authorised to give the instruction set out in section 3.3.1 on behalf of each relevant Company Affiliate.

3.4 Annex 1 to this Addendum sets out certain information regarding the Contracted Processors' Processing of the Company Personal Data as required by article 28(3) of the GDPR. Company may make reasonable amendments to Annex 1 by written notice to Vendor from time to time as Company reasonably considers necessary to meet those requirements. Nothing in Annex 1 (including as amended pursuant to this section 3. 4) confers any right or imposes any obligation on any party to this DPA.

4. Vendor and Vendor Affiliate Personnel

Vendor and each Vendor Affiliate shall take reasonable steps to ensure the reliability of any employee, agent or contractor of any Contracted Processor who may have access to the Company Personal Data, ensuring in each case that access is strictly limited to

those individuals who need to know the relevant Company Personal Data, as strictly necessary for the purposes of the Services and consistent with the Principal Contracts, and to comply with Applicable Laws in the context of that individual's duties to the Contracted Processor, ensuring that all such individuals are subject to confidentiality undertakings or professional or statutory obligations of confidentiality.

5. Security

5.1 Vendor and Vendor Affiliates that perform any Processing of Customer Data shall implement and maintain appropriate technical and organizational measures for protection of the security, confidentiality and integrity of Company Personal Data, including measures that protect against unauthorized or unlawful Processing and against accidental or unlawful destruction, loss or alteration or damage, unauthorized disclosure of, or access to, Company Personal Data and, as appropriate, the measures referred to in Article 32(1) of the GDPR.

5.2 Vendor and Vendor Affiliates that perform any Processing of Customer Data shall undergo annual third-party Type II SSAE 16 audits (or equivalent audits) at Vendor's expense. Upon Company's written request at reasonable intervals, and subject to the confidentiality obligations set forth in any Principal Contract, Vendor shall make available to Company a copy of the most recent third-party audit report.

6. Subprocessing

6.1 Each Company Group Member authorises Vendor and each Vendor Affiliate to appoint (and permit each Subprocessor appointed in accordance with this section 6 to appoint) Subprocessors in accordance with this section 6 and any restrictions in the Principal Contracts.

6.2 Vendor and each Vendor Affiliate may continue to use those Subprocessors already engaged by Vendor or any Vendor Affiliate as of the date of this DPA, subject to Vendor and each Vendor Affiliate in each case as soon as practicable meeting the obligations set out in section 6.4

6.3 Vendor shall give Company prior written notice of the appointment of any new Subprocessor, including full details of the Processing to be undertaken by the Subprocessor. If, within thirty (30) days of receipt of that notice, Company notifies Vendor in writing of any objections (on reasonable grounds) to the proposed appointment:

6.3.1 Vendor shall work with Company in good faith to make available a commercially reasonable change in the provision of the Services which avoids the use of that proposed Subprocessor; and

6.3.2 where such a change cannot be made within thirty (30) days from Vendor's receipt of Company's notice, notwithstanding anything in the Principal Contracts, Company may by written notice to Vendor with immediate effect terminate the Principal Contracts.

6.4 With respect to each Subprocessor, Vendor or the relevant Vendor Affiliate shall:

6.4.1 before the Subprocessor first Processes Company Personal Data (or, where relevant, in accordance with section 6.2), carry out adequate due diligence to ensure that the Subprocessor is capable of providing adequate protection for Company Personal Data;

6.4.2 ensure that the arrangement between on the one hand (a) Vendor, or (b) the relevant Vendor Affiliate, or (c) the relevant intermediate Subprocessor; and on the other hand the Subprocessor, is governed by a written contract including terms which offer at least the same level of protection for Company Personal Data as those set out in this DPA and meet the requirements of article 28(3) of the GDPR;

6.4.3 if that arrangement involves a Restricted Transfer, ensure that the Standard Contractual Clauses are at all relevant times incorporated into the agreement between on the one hand (a) Vendor, or (b) the relevant Vendor Affiliate, or (c) the relevant intermediate Subprocessor; and on the other hand the Subprocessor; and

6.4.4 provide to Company for review such copies of the Contracted Processors' agreements with Subprocessors (which may be redacted to remove confidential commercial information not relevant to the requirements of this DPA) as Company may request from time to time.

6.5 Vendor and each Vendor Affiliate shall ensure that each Subprocessor performs the obligations under sections 3.2, 4, 5, 7.1, 8.2, 9 and 11, as they apply to Processing of Company Personal Data carried out by that Subprocessor, as if it were party to this DPA in place of Vendor.

7. Data Subject Rights

7.1 Taking into account the nature of the Processing, Vendor and each Vendor Affiliate shall assist each Company Group Member by implementing appropriate technical and organisational measures, insofar as this is possible, for the fulfilment of the Company Group Members' obligations, as reasonably understood by Company, to respond to requests to exercise Data Subject rights under the Data Protection Laws.

7.2 Vendor shall:

7.2.1 promptly notify Company if any Contracted Processor receives a request from a Data Subject under any Data Protection Law in respect of Company Personal Data; and

7.2.2 ensure that the Contracted Processor does not respond to that request except on the documented instructions of Company or the relevant Company Affiliate or as required by Applicable Laws to which the Contracted Processor is subject, in which case Vendor shall to the extent permitted by Applicable Laws inform Company of that legal requirement before the Contracted Processor responds to the request.

8. Personal Data Breach

8.1 Vendor shall notify Company without undue delay upon Vendor or any Subprocessor becoming aware of a Personal Data Breach affecting Company Personal Data, providing Company with sufficient information to allow each Company Group Member to meet any obligations to report or inform Data Subjects of the Personal Data Breach under the Data Protection Laws.

8.2 Vendor shall co-operate with Company and each Company Group Member and take such reasonable commercial steps to assist in the investigation, mitigation and remediation of each such Personal Data Breach.

9.Data Protection Impact Assessment and Prior Consultation

Vendor and each Vendor Affiliate shall provide reasonable assistance to each Company Group Member with any data protection impact assessments, and prior consultations with Supervising Authorities or other competent data privacy authorities, which Company reasonably considers to be required of any Company Group Member by article 35 or 36 of the GDPR or equivalent provisions of any other Data Protection Law, in each case solely in relation to Processing of Company Personal Data by, and taking into account the nature of the Processing and information available to, the Contracted Processors.

10.Deletion of Company Personal Data

Vendor and each Vendor Affiliate shall promptly and in any event within thirty (30) days after the date of termination of Services involving the Processing of Company Personal Data, delete such Company Personal Data and all copies thereof, so that they cannot be recovered or reconstructed. Vendor shall provide written certification to Company that it and each Vendor Affiliate has fully complied with this section 10.

11.Audit rights

11.1 Subject to sections [11.2 to 11.4], Vendor and each Vendor Affiliate shall make available to each Company Group Member on request all information necessary to demonstrate compliance with this Addendum, and shall allow for and contribute to audits, including inspections, by any Company Group Member or an auditor mandated by any Company Group Member in relation to the Processing of the Company Personal Data by the Contracted Processors.

11.2 Information and audit rights of the Company Group Members only arise under section 11.1 to the extent that the Principal Agreement does not otherwise give them information and audit rights meeting the relevant requirements of Data Protection Law (including, where applicable, article 28(3)(h) of the GDPR).

11.3 Company or the relevant Company Affiliate undertaking an audit shall give Vendor or the relevant Vendor Affiliate reasonable notice of any audit or inspection to be conducted under section 11.1 and shall make (and ensure that each of its mandated auditors makes) reasonable endeavours to avoid causing (or, if it cannot avoid, to minimise) any damage, injury or disruption to the Contracted Processors' premises, equipment, personnel and business while its personnel are on those premises in the course of such an audit or inspection. A Contracted Processor need not give access to its premises for the purposes of such an audit or inspection:

11.3.1 to any individual unless he or she produces reasonable evidence of identity and authority;

11.3.2 outside normal business hours at those premises, unless the audit or inspection needs to be conducted on an emergency basis and Company or the relevant Company Affiliate undertaking an audit has given notice to Vendor or the relevant Vendor Affiliate that this is the case before attendance outside those hours begins; or

11.3.3 for the purposes of more than [one] audit or inspection, in respect of each Contracted Processor, in any [calendar year], except for any additional audits or inspections which:

11.3.3.1 Company or the relevant Company Affiliate undertaking an audit reasonably considers necessary because of genuine concerns as to Vendor's or the relevant Vendor Affiliate's compliance with this Addendum; or

11.3.3.2 A Company Group Member is required or requested to carry out by Data Protection Law, a Supervisory Authority or any similar regulatory authority responsible for the enforcement of Data Protection Laws in any country or territory,

where Company or the relevant Company Affiliate undertaking an audit has identified its concerns or the relevant requirement or request in its notice to Vendor or the relevant Vendor Affiliate of the audit or inspection.]

12. Restricted Transfers

12.1 Subject to section 12.2, each Company Group Member (as "data exporter") and each Contracted Processor, as appropriate, (as "data importer") hereby enter into the Standard Contractual Clauses in respect of any Restricted Transfer from that Company Group Member to that Contracted Processor.

12.2 Section 12.1 shall not apply to a Restricted Transfer unless its effect, together with other reasonably practicable compliance steps (which, for the avoidance of doubt, do not include obtaining consents from Data Subjects), is to allow the relevant Restricted Transfer to take place without breach of applicable Data Protection Law.

12.3 Vendor warrants and represents that, before the commencement of any Restricted Transfer to a Subprocessor which is not a Vendor Affiliate, Vendor's or the relevant Vendor Affiliate's entry into the Standard Contractual Clauses under section 12.1, and variations to those Standard Contractual Clauses made under section 13.4, as agent for and on behalf of that Subprocessor will have been duly and effectively authorised (or subsequently ratified) by that Subprocessor.

13. General Terms

Governing law and jurisdiction

13.1 Without prejudice to clauses 7 (Mediation and Jurisdiction) and 9 (Governing Law) of the Standard Contractual Clauses:

13.1.1 the parties to this DPA hereby submit to the choice of jurisdiction stipulated in the Principal Contracts with respect to any disputes or claims howsoever arising under this DPA, including disputes regarding its existence, validity or termination or the consequences of its nullity; and

13.1.2 this DPA and all non-contractual or other obligations arising out of or in connection with it are governed by the laws of the country or territory stipulated for this purpose in the Principal Contracts.

Order of precedence

13.2 Nothing in this DPA reduces Vendor's or any Vendor Affiliate's obligations under the Principal Contracts in relation to the protection of Personal Data or permits Vendor or any Vendor Affiliate to Process (or permit the Processing of) Personal Data in a manner which is prohibited by the Principal Contracts. In the event of any conflict or inconsistency between this DPA and the Standard Contractual Clauses, the Standard Contractual Clauses shall prevail.

13.3 Subject to section 13.2, with regard to the subject matter of this DPA, in the event of inconsistencies between the provisions of this DPA and any other agreements between the parties, including the Principal Contracts and including (except where explicitly agreed otherwise in writing, signed on behalf of the parties) agreements entered into or purported to be entered into after the date of this DPA, the provisions of this DPA shall prevail.

Changes in Data Protection Laws, etc.

13.4 Company may, by at least 30 (thirty)-days' written notice to Vendor from time to time, make any variations to the Standard Contractual Clauses (including any Standard Contractual Clauses entered into under section 12.1), as they apply to Restricted Transfers which are subject to a particular Data Protection Law, which are required, as a result of any change in, or decision of a

competent authority under, that Data Protection Law, to allow those Restricted Transfers to be made (or continue to be made) without breach of that Data Protection Law.

13.5 If Company gives notice under section 13.4, Vendor and each Vendor Affiliate shall promptly co-operate (and ensure that any affected Subprocessors promptly co-operate) to ensure that equivalent variations are made to any agreement put in place under section 6.4.3.

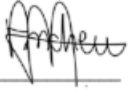
13.6 Neither Company nor Vendor shall require the consent or approval of any Company Affiliate or Vendor Affiliate to amend this DPA pursuant to section 13.5 or otherwise.

Severance

13.7 Should any provision of this DPA be invalid or unenforceable, then the remainder of this DPA shall remain valid and in force. The invalid or unenforceable provision shall be either (i) amended as necessary to ensure its validity and enforceability, while preserving the parties' intentions as closely as possible or, if this is not possible, (ii) construed in a manner as if the invalid or unenforceable part had never been contained therein.

IN WITNESS WHEREOF, this DPA is entered into and becomes a binding part of the Principal Contracts with effect from the date first set out above.

Company: Crowditivity

Signature 

Name Andrew Arnita

Title Identifying Annotation Types

Date Signed 28/05/2023

Vendor: QuestionPro INC

Signature 
VIVEK BHASKARAN

Name: Vivek Bhaskaran

Title: CEO

Date Signed: 28/05/2023

Appendix G

User Annotations from Case Study

This appendix regroupes all the annotated images created by the participants involved in the case study.



Figure G.1: Image Annotations Participant 1

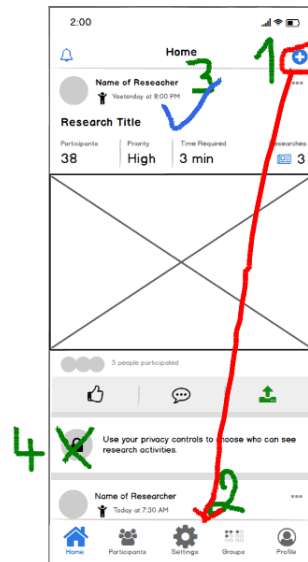


Figure G.2: Image Annotations Participant 2

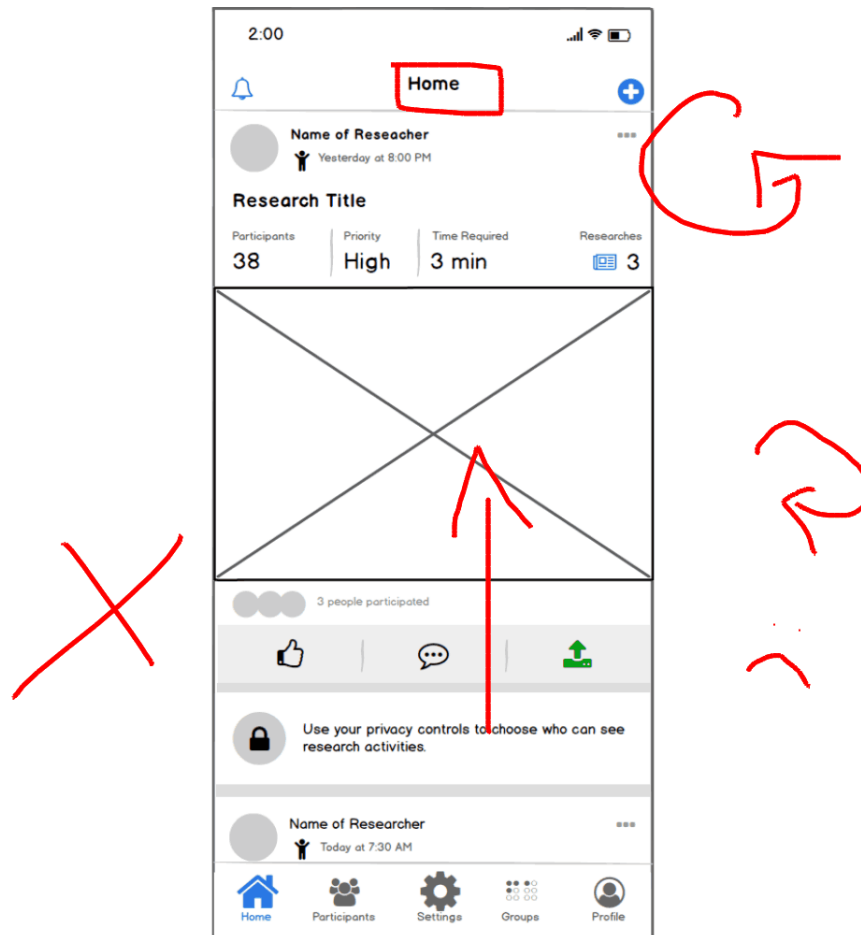


Figure G.3: Image Annotations Participant 3

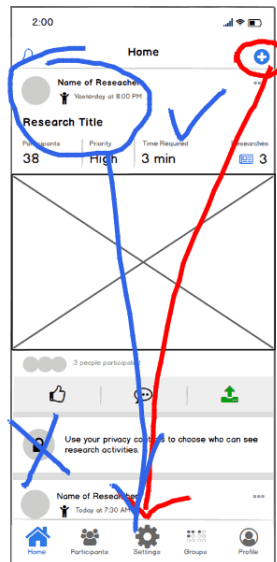


Figure G.4: Image Annotations Participant 4



Figure G.5: Image Annotations Participant 5



Figure G.6: Image Annotations Participant 6

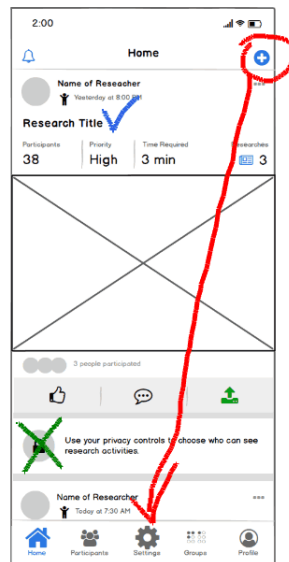


Figure G.7: Image Annotations Participant 7



Figure G.8: Image Annotations Participant 8

Glossary

.jpeg	Extension for image file.
.jpg	Extension for image file.
.pdf	Extension for document file.
.png	Extension for image file.
.svg	Extension for image file.
AGATe	AGreement Analysis Toolkit.
AR	Agreement Rate.
Crowditivity	Crowd Activity.
Crowdlicit	Crowd Elicitation.
Crowdlicit	Crowd Consensus.
GES	Gesture elicitation study.
RE	Requirements Engineering.
Recognition Time	Time required by a recognizer to classify an annotation, expressed here in milliseconds.
Sampling Points	Refers to a set of coordinates to give a visual representation of an annotation.
SDLC	Software Development Life Cycle.
UCLouvain	Universite Catholique de Louvain.
UEQ+	User Experience Questionnaire Plus.
UI	User Interface.
UX	User Experience.

Bibliography

- [1] Annette Adler, Anuj Gujar, Beverly Harrison, Kenton O’Hara, and Abigail Sellen. A diary study of work-related reading: Design implications for digital reading devices. pages 241–248, 01 1998.
- [2] M J Adler and C Van Doren. *How to read a book : the classic guide to intelligent reading*. Simon and Schuster, 1972.
- [3] Abdullah X. Ali, Meredith Ringel Morris, and Jacob O. Wobbrock. Crowdsourcing similarity judgments for agreement analysis in end-user elicitation studies. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST ’18, page 177–188, New York, NY, USA, 2018. Association for Computing Machinery.
- [4] Abdullah X. Ali, Meredith Ringel Morris, and Jacob O. Wobbrock. Crowdlicit: A system for conducting distributed end-user elicitation and identification studies. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI ’19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.
- [5] Christine Alvarado, Andy Kearney, Alexa Keizur, Calvin Loncaric, Miranda Parker, Jessica Peck, Kiley Sobel, and Fiona Tay. *LogiSketch: A Free-Sketch Digital Circuit Design and Simulation System* *LogiSketch*, pages 83–90. 07 2015.
- [6] Lisa Anthony and Jacob O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI ’10, page 245–252, CAN, 2010. Canadian Information Processing Society.
- [7] Daniel Bildhauer, Tassilo Horn, and Jurgen Ebert. Similarity-driven software reuse. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models*, CVSM ’09, page 31–36, USA, 2009. IEEE Computer Society.
- [8] Azita Soltanian Bojnord. A support tool for elicitation of user interface requirements. 2011.
- [9] Azita Soltanian Bojnord. A support tool for elicitation of user interface requirements. 2011.
- [10] Matthias Kevin Caspers. React and redux. *Rich Internet Applications w/HTML and Javascript*, 11, 2017.
- [11] Peter de Lange, Petru Nicolaescu, Alexander Tobias Neumann, and Ralf Klamma. Integrating web-based collaborative live editing and wireframing into a model-driven web engineering process. *Data Science and Engineering*, 5(3):240–260, Sep 2020.

- [12] Jodi Forlizzi and Katja Battarbee. Understanding experience in interactive systems. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, page 261–268, New York, NY, USA, 2004. Association for Computing Machinery.
- [13] Robert L H Fowler and Anne S. Barker. Effectiveness of highlighting for retention of text material. *Journal of Applied Psychology*, 59:358–364, 1974.
- [14] Julián Grigera, José Matías Rivero, Esteban Robles Luna, Franco Giacosa, and Gustavo Rossi. From requirements to web applications in an agile model-driven approach. In *International Conference on Web Engineering*, 2012.
- [15] Jean-Luc Hak. *Engineering annotations for supporting the design process of interactive systems: A model-based approach and a tool suite*. PhD thesis, 07 2019.
- [16] Marc Hassenzahl and Noam Tractinsky. User experience - a research agenda. *Behaviour and Information Technology*, 25:91 – 97, 03 2006.
- [17] Chris Hersman and Kim Fowler. Chapter 5 - best practices in spacecraft development. In Kim Fowler, editor, *Mission-Critical and Safety-Critical Systems Handbook*, pages 269–460. Newnes, Boston, 2010.
- [18] Andreas Hinderks, Dominique Winter, Francisco José Domínguez Mayo, M.J. Escalona, and Jörg Thomaschewski. Ux poker: Estimating the influence of user stories on user experience in early stage of agile development. *International Journal of Interactive Multimedia and Artificial Intelligence*, 7:97, 12 2022.
- [19] Md Athar Imtiaz, Rachel Blagojevic, Andrew Luxton-Reilly, and Beryl Plimmer. A survey of intelligent digital ink tools use in stem education. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '17, New York, NY, USA, 2017. Association for Computing Machinery.
- [20] Arshad Javeed. Performance optimization techniques for reactjs. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–5, 2019.
- [21] Atif A. Jilani, Muhammad Z. Iqbal, Muhammad U. Khan, and Muhammad Usman. Chapter three - advances in applications of object constraint language for software engineering. volume 112 of *Advances in Computers*, pages 135–184. Elsevier, 2019.
- [22] Audris Kalnins, Tomasz Straszak, Michał Śmiałek, Elina Kalnina, Edgars Celms, and Wiktor Nowakowski. Developing software with domain-driven model reuse. In *Handbook of Research on Innovations in Systems and Software Engineering*, pages 283–312. IGI Global, 2015.
- [23] Ricardo Kawase, Eelco Herder, and Wolfgang Nejdl. A comparison of paper-based and online annotations in the workplace. pages 240–253, 09 2009.
- [24] Amal Khalil and Juergen Dingel. Chapter four - optimizing the symbolic execution of evolving rhapsody statecharts. volume 108 of *Advances in Computers*, pages 145–281. Elsevier, 2018.

- [25] Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. User-defined gestures for connecting mobile phones, public displays, and tabletops. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, page 239–248, New York, NY, USA, 2010. Association for Computing Machinery.
- [26] Sujit Kumar and Pushkar Dubey. Software development life cycle (sdlc) analytical comparison and survey on traditional and agile methodology. *ABHINAV NATIONAL MONTHLY REFEREED JOURNAL OF RESEARCH IN SCIENCE TECHNOLOGY*, 2:22–30, 08 2013.
- [27] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. volume 5298, pages 63–76, 11 2008.
- [28] Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan. Software development life cycle agile vs traditional approaches. 37(1):162–167, 2012.
- [29] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [30] Nathan Magrofuoco and Jean Vanderdonckt. Gelicit: A cloud platform for distributed gesture elicitation studies. *Proc. ACM Hum.-Comput. Interact.*, 3(EICS), jun 2019.
- [31] Jean MaIm, Federico Ciccozzi, Jan Gustafsson, Björn Lisper, and Jonas Skoog. Static flow analysis of the action language for foundational uml. pages 161–168, 09 2018.
- [32] Chris Porter, Emmanuel Letier, and M. Angela Sasse. Building a national e-service using sentire experience report on the use of sentire: A volere-based requirements framework driven by calibrated personas and simulated user feedback. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 374–383, 2014.
- [33] Thiruvengadam Radhakrishnan, Ahmed Seffah, and Asmaa Alsumait. User interface requirements engineering: a scenario-based framework. 2004.
- [34] José Matías Rivero, Gustavo Rossi, Julián Grigera, Juan Burella, Esteban Robles Luna, and Silvia Gordillo. From mockups to user interface models: An extensible model driven approach. In Florian Daniel and Federico Michele Facca, editors, *Current Trends in Web Engineering*, pages 13–24, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [35] J Rosenfeld L Morville P Arango. *Information Architecture : For the Web and Beyond*. Sebastopol CA: O'Reilly Media, 2015.
- [36] Xavier Ryckel, Arthur Sluÿters, and Jean Vanderdonckt. Snappview, a software development kit for supporting end-user mobile interface review. *Proceedings of the ACM on Human-Computer Interaction*, 6:1–38, 06 2022.
- [37] Harry Santoso, Martin Schrepp, Lintang Hasani, Rian Fitriansyah, and Arief Setyanto. The use of user experience questionnaire plus (ueq+) for cross-cultural ux research: evaluating zoom and learn quran tajwid as online learning tools. *Heliyon*, 8:e11748, 11 2022.
- [38] Martin Schrepp. User experience questionnaire handbook. 09 2015.

- [39] Martin Schrepp. Measuring user experience with modular questionnaires. In *2021 International Conference on Advanced Computer Science and Information Systems (ICACISIS)*, pages 1–6, 2021.
- [40] Martin Schrepp and Jörg Thomaschewski. Design and validation of a framework for the creation of user experience questionnaires. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5(7):88–95, 12/2019 2019.
- [41] Maximilian Speicher and Michael Nebeling. Gesturewiz: A human-powered gesture design environment for user interface prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–11, New York, NY, USA, 2018. Association for Computing Machinery.
- [42] Eccam s.r.o. Volere requirements specification template, NaN.
- [43] Alistair Sutcliffe and Jan Gulliksen. Chapter 18 - user-centered requirements definition. In Elizabeth Buie and Dianne Murray, editors, *Usability in Government Systems*, pages 285–300. Morgan Kaufmann, Boston, 2012.
- [44] Craig Sutherland, Andrew Luxton-Reilly, and Beryl Plimmer. Freeform digital ink annotations in electronic documents: A systematic mapping study. *Computers Graphics*, 55, 11 2015.
- [45] T.H. Tse and L. Pong. An examination of requirements specification languages. *The Computer Journal*, 34, 07 2003.
- [46] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. !ftl, an articulation-invariant stroke gesture recognizer with controllable position, scale, and rotation invariances. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ICMI '18, page 125–134, New York, NY, USA, 2018. Association for Computing Machinery.
- [47] Radu-Daniel Vatavu. Improving gesture recognition accuracy on touch screens for users with low vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4667–4679, New York, NY, USA, 2017. Association for Computing Machinery.
- [48] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. Gestures as point clouds: A \$p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, page 273–280, New York, NY, USA, 2012. Association for Computing Machinery.
- [49] Radu-Daniel Vatavu and Jacob O. Wobbrock. Formalizing agreement analysis for elicitation studies: New measures, significance test, and toolkit. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 1325–1334, New York, NY, USA, 2015. Association for Computing Machinery.
- [50] Radu-Daniel Vatavu and Jacob O. Wobbrock. Between-subjects elicitation studies: Formalization and tool support. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 3390–3402, New York, NY, USA, 2016. Association for Computing Machinery.

- [51] Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A. Myers. Maximizing the guessability of symbolic input. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, page 1869–1872, New York, NY, USA, 2005. Association for Computing Machinery.
- [52] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 1083–1092, New York, NY, USA, 2009. Association for Computing Machinery.
- [53] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: A \$1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, page 159–168, New York, NY, USA, 2007. Association for Computing Machinery.
- [54] Michał Śmiałek and Tomasz Straszak. Facilitating transition from requirements to code with the redseeds tool. pages 321–322, 09 2012.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl