



UNIVERSIDADE D
COIMBRA

Diogo Malheiro Boinas

**A Software Tool for Customer Experience
Evaluation in Service Design
MobEthnos**

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Paulo Rupino da Cunha and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2022

Faculty of Sciences and Technology
Department of Informatics Engineering

A Software Tool for Customer Experience Evaluation in Service Design

MobEthnos

Diogo Malheiro Boinas

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Paulo Rupino da Cunha and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022



UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Abstract

In this thesis it was proposed to build a software tool with the objective of evaluating services. This tool consists of a Progressive Web Application (PWA) for service consumers and a website for service providers. React and React Native frameworks were used for development. All phases of a software engineering project have been thought out and executed. Besides the software tool, other important artifacts were generated, such as a major study about the subject which covered the topics Mobile Ethnography and Service Design, a requirements document, a software architecture and several test cases.

From the PWA, feedback is collected from service consumers using Mobile Ethnography as a data collection method. It works like a native mobile application and allows offline use. In this way, users can submit their experience in real time and in the service environment, despite not having an internet connection. These two are objectives of Mobile Ethnography.

The website uses Service Design guidelines to be able to evaluate the service fairly. Thus, it is possible to make, for example, custom journey maps for each user, obtain a map that dictates the best and worst areas of a service and view some graphs about the service and consumer actions. This is a great help for service providers as they have the right tools to carry out the evaluation.

By using the software tool, it is possible to improve services by detecting their weaknesses due to the data being rich, this is having photos, videos and other important information. As this tool is for private use between the customer and the service provider, better transparency is achieved.

Keywords

Mobile Ethnography; Service Design; Customer Experience Evaluation; Web Development; Mobile Development.

This page is intentionally left blank.

Resumo

Nesta tese foi proposto construir uma ferramenta de software com o objetivo de avaliar serviços. Esta ferramenta consiste numa aplicação web progressiva para consumidores de serviços e um website para fornecedores de serviços. Foram utilizadas as frameworks React e React Native para o desenvolvimento. Todas as fases de um projeto de engenharia de software foram pensadas e executadas. Além da ferramenta de software, outros artefatos importantes foram gerados, tais como um grande estudo sobre o tema que incluiu os temas Etnografia Móvel e Design de Serviço, um documento de requisitos, uma arquitetura de software e diversos casos de teste.

A partir da aplicação web progressiva, recolhe-se feedback dos consumidores do serviço utilizando como método de recolha de dados etnografia móvel. Funciona como uma aplicação móvel nativa e permite uma utilização offline. Dessa forma, os utilizadores podem enviar a sua experiência em tempo real e no ambiente do serviço, mesmo não existindo uma conexão com a internet. Estes dois são objetivos da etnografia móvel.

O website usa diretrizes de design de serviço para poder avaliar o serviço de forma equitativa. Desta forma, é possível fazer, por exemplo, customer journey maps para cada utilizador, obter um mapa que dita as melhores e piores áreas de um serviço e visualizar alguns gráficos sobre o serviço e ações do consumidor. Esta é uma grande ajuda para os fornecedores de serviços, porque assim têm as ferramentas certas para fazer a avaliação.

Com uso da ferramenta de software consegue-se melhorar serviços ao detatar os seus pontos fracos pelo facto de os dados serem ricos, isto é, incluir fotos, vídeos e outra informação importante. Como esta ferramenta é de uso privado entre o consumidor e o fornecedor do serviço atinge-se uma melhor transparência.

Palavras-Chave

Etnografia Móvel; Design de Serviço; Avaliação da Experiência do Consumidor; Desenvolvimento Web; Desenvolvimento Móvel.

This page is intentionally left blank.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	2
1.3	Software Development Process and Planning	2
1.4	Report Structure	8
2	State-Of-The-Art	9
2.1	State-Of-The-Art Procedure	9
2.2	Service Design	10
2.3	Mobile Ethnography	11
2.4	Mobile Ethnography Software Tools	12
2.5	Technologies to help build the Software Tool	18
2.5.1	Website Front-end	18
2.5.2	Mobile Application Front-end	20
2.5.3	Back-end	21
2.5.4	Hosting	23
3	Requirements	25
3.1	Requirements Gathering	25
3.2	Use Case Model	26
3.3	User Interface Model	30
3.4	Non-Functional Requirements	34
3.5	Functional Requirements Prioritisation	35
3.5.1	Participant Requirements	35
3.5.2	Researcher Requirements	35
4	Software Architecture	37
5	Software Development	45
5.1	Back-end Structure	46
5.1.1	Firebase Authentication	46
5.1.2	Cloud Firestore	46
5.1.3	Firebase Storage	47
5.1.4	Security	48
5.2	Mobile Application Development	48
5.2.1	Creation of Files and Installation of Modules	48
5.2.2	Connection to Firebase	49
5.2.3	Authentication and Navigation	50
5.2.4	Unauthenticated User Functionalities	52
5.2.5	Authenticated User Functionalities	53
5.2.6	Progressive Web App Requirements	59
5.3	Website Development	60

5.3.1	Creation of Files and Installation of Modules	60
5.3.2	Connection to Firebase	60
5.3.3	Navigation Between Components	60
5.3.4	Authentication	61
5.3.5	Unauthenticated User Functionalities	61
5.3.6	Authenticated User Functionalities	62
6	Software Testing	67
6.1	End-To-end Testing	67
6.2	Usability Testing	68
6.2.1	Participants	69
6.2.2	Procedure	69
6.2.3	Results	69
6.2.4	Conclusions	74
6.3	Software Testing Conclusions	74
7	Conclusion	75
	References	78
	Appendix A - Use Cases	83
	Appendix B - User Interface	99
	Appendix C - Testing Results and Descriptions	112

This page is intentionally left blank.

Acronyms

Amazon RDS Amazon Relational Database Service. 22

Amazon S3 Amazon Simple Storage Service. 21, 22

API Application Programming Interface. 21, 23

AWS Amazon Web Services. 9, 21, 22, 23

DEI Department of Informatics Engineering. 20, 23, 45, 75, 76

DOM Document Object Model. 18, 19

GDP Gross Domestic Product. 1, 10

PWA Progressive Web Application. iii, 20, 21, 23, 45, 59, 61, 67, 75, 76, 77

SDK Software Development Kit. 19, 20, 21, 23, 48, 59

UED User Experience Diagram. 26, 30, 31, 33, 76

UI User Interface. 18, 26, 30, 31, 33, 34, 76

UML Unified Modeling Language. 43

VPS Virtual Private Server. 23

This page is intentionally left blank.

Glossary

touchpoint is an interaction of a user with a service and includes several information that helps in qualitative research, such as title, emotion or satisfaction, notes, media, location, time, and date. As mobile ethnography is user-centered, each user has several touchpoints regarding a specific service. . xiv, 12, 13, 14, 15, 16, 17, 20, 30, 33, 35, 46, 47, 51, 53, 54, 55, 56, 57, 58, 59, 62, 63, 64, 65, 69, 70, 73, 74

This page is intentionally left blank.

List of Figures

1.1	First Semester Gantt Chart Estimation	4
1.2	Actual First Semester Gantt Chart	5
1.3	Second Semester Gantt Chart Estimation	6
1.4	Actual Second Semester Gantt Chart	7
2.1	Inside the project page in the Mobile Application. Figure taken from Annex 1.	13
2.2	Inside the create new touch point page in the Mobile Application. Figure taken from Annex 1.	13
2.3	Example of the QR code to invite participants	14
2.4	Example of the Raw Data functionality.	15
2.5	Example of the Raw Data functionality.	15
2.6	Example of the overview functionality.	16
2.7	Example of the map with the touchpoints	17
3.1	Architecture Sketch	26
3.2	Context Diagram.	27
3.3	Use Cases Diagram for Signing In and Signing Up.	27
3.4	Use Cases Diagram for Participant Functionalities.	28
3.5	Use Cases Diagram for Researcher Functionalities.	29
3.6	User Experience Diagram of Website	31
3.7	User Experience Diagram of Mobile Application	32
3.8	Homepage User Interface	33
3.9	Homepage User Interface.	34
3.10	Project User Interface.	34
4.1	Context Diagram	38
4.2	Container Diagram	39
4.3	Progressive Web Application Component Diagram	41
4.4	Single-Page Application Component Diagram	42
5.1	Cloud Firestore Database Schema	47
5.2	Firebase Storage Folder Structure	48
5.3	Register Screen of the Mobile Application	53
5.4	Projects Screen of the Mobile Application	54
5.5	Project Screen of the Mobile Application	56
5.6	Add Touchpoint Screen of the Mobile Application	57
5.7	Successful Internet Connection Notification	57
5.8	Unsuccessful Internet Connection Notification	57
5.9	Structuring of Data for the Touchpoints Array	64
6.1	Log-in test using Selenium IDE	68

6.2	Question one	71
6.3	Question two	71
6.4	Question three	72
6.5	Question four	72
6.6	Question five	73
1	Sign Up User Interface	99
2	Dashboard User Interface	100
3	Create Project User Interface	101
4	Project Created User Interface	102
5	View Projects User Interface	103
6	Project Journeys User Interface	104
7	Project Map User Interface	105
8	Project Data User Interface	106
9	Project Overview User Interface	107
10	Sign In Interface.	108
11	Sign Up User Interface.	108
12	Settings Interface.	109
13	Add Project User Interface.	109
14	Choose Project User Interface.	110
15	New/Edit Touchpoint User Interface.	110

This page is intentionally left blank.

List of Tables

3.1	Participant Sign Up Use Case Table	30
6.1	Metrics for each user for each completed task	70
6.2	Expected Number of Clicks for Each Task	70
6.3	Mean Number of Clicks for Each Task	70
1	Researcher Sign Up Use Case Table	83
2	Participant Sign In Use Case Table	84
3	Researcher Sign In Use Case Table	85
4	Join a New Project Use Case Table	86
5	Views Project that you are Involved In Use Case Table	86
6	Change Password Use Case Table	87
7	Delete Account Use Case Table	87
8	View all touchpoint points submitted Use Case Table	88
9	Edit Touchpoint Use Case Table	88
10	Edit Touchpoint Use Case Table	89
11	Leave Project Use Case Table	90
12	Participant’s Logout Use Case Table	90
13	Change Password Use Case Table	91
14	Delete Account Use Case Table	91
15	Create Project Use Case Table	92
16	View Own Projects Use Case Table	92
17	Finalize Projects Use Case Table	93
18	Finalize Projects Use Case Table	93
19	View Project Journeys Use Case Table	94
20	Export Data Use Case Table	94
21	Manipulate Project Journeys Use Case Table	95
22	Delete Project Use Case Table	96
23	Show Project Map Use Case Table	96
24	Show Overview Graph Use Case Table	97
25	Researcher’s Logout Use Case Table	97
26	End-to-end testing conclusions	112
27	Description of Test with ID 1	112
28	Description of Test with ID 2	113
29	Description of Test with ID 3	113
30	Description of Test with ID 4	113
31	Description of Test with ID 5	113
32	Description of Test with ID 6	113
33	Description of Test with ID 7	114
34	Description of Test with ID 8	114
35	Description of Test with ID 9	114

36	Description of Test with ID 10	114
37	Description of Test with ID 11	114
38	Description of Test with ID 12	115
39	Description of Test with ID 13	115
40	Description of Test with ID 14	115
41	Description of Test with ID 15	115
42	Description of Test with ID 16	115
43	Description of Test with ID 17	115
44	Description of Test with ID 18	116

This page is intentionally left blank.

Chapter 1

Introduction

1.1 Motivation

We live in a world where services represent about 65% of Gross Domestic Product (GDP) according to data from the world bank [6]. GDP is the monetary value of all finished products and services produced within a country during a certain period of time [20]. Also, according to these data [6], we can see that it is a number that is always increasing. We have data from 1995 to 2020. By analysing the world bank chart [6] we can reach the conclusion that the need for services in the current world is indispensable. We can also see that in developed countries this number is much higher. According to the world bank, in the United States, the GDP of services is around 77% [7].

According to the book *Ethnography*, their definition for ethnography is "(...) the study of people in naturally occurring settings or 'fields' by means of methods which capture their social meanings and ordinary activities, involving the researcher participating directly in the setting, if not also the activities, in order to collect data in a systematic manner but without meaning being imposed on them externally." [8].

Mobile ethnography "(...) uses technology-based devices, e.g. smartphones, instead of traditional ethnographic face-to-face inquiry." [32]. It is based on collecting data using mobile devices; in this way larger amounts of data can be collected, as opposed to classical ethnography, which involves having a researcher with the participant directly. This is much more costly than using a mobile device. In addition, more reliable data can be collected as the participant does it with a certain anonymity and by himself on his mobile device.

In this dissertation, our main motivation is to apply Mobile Ethnography to the Mobile Application and to apply service design to the website. By having a Mobile Ethnography tool that allows customers to express their opinions about a service, researchers can help to increase the quality of that specific service by using the service design guidelines. With a software like this, we hope to be able to collect larger amounts and more reliable data. Another topic of this software is to have everything incorporated in one platform, thus allowing the researcher to analyse the collected data without having to export it to another platform.

1.2 Objectives

The main objective in this dissertation is to develop a software tool for customer experience evaluation. We do customer experience evaluation by using mobile ethnography. Also, this software is related to service design, in the way that our objective is to help improving services.

To start this dissertation, a good understanding of service design will be needed. After learning about this theme, we want to learn about Mobile Ethnography. First, we need to understand what it is. Then, we need to look for existing tools to help us gather the requirements for our platform.

This platform will include a website application for researchers and a mobile application for participants. On the one hand, the mobile application will allow participants to send data and must work on the Android and iOS operating systems. On the other hand, the website application will allow for the researcher to analyse the collected data and it has to work in web browsers.

1.3 Software Development Process and Planning

During the development of the software tool, the methodology to be used will be based on Scrum [42]. The work to be done will all be made in sprints for the tasks completion. Scrum is good for teams with a small number of elements, less than 10 participants.

In our case, there is not exactly a development team. This will be done by the author of this thesis as he is the one to do all the development. We can also say that the Product Owner will be the advisor of this dissertation, Prof. Paulo Rupino da Cunha, as he was the one who came up with this project. The main role of a Product Owner is to create an interaction between the customer and the Development Team [42].

Regarding the main artifacts that represent the Scrum methodology, there exists the Product Backlog, the Sprint Backlog and the Product Increment [33]. There also exist other artifacts such as release burndown charts and release plan, however, they are not as important as the ones described in the previous sentence [33].

In our particular case, we can have a similar artifact to the Product Backlog, which will be the software requirements specification document. It will contain all the information about the requirements to be implemented in the software development phase. Typically, the Product Backlog contains a list of requirements given by the client and can be changed as new requirements apply. The element of the team that can change this artifact is the Product Owner.

Regarding the Sprint Backlog, the tasks will be done in a similar manner, choosing a set of requirements to be implemented in the following sprint and then testing it and see if it is alright to go to the next one. What the Sprint Backlog represents is a list of requirements to be done in the next sprint.

Another part of scrum are the meetings; after each sprint, there is a meeting to assess what has been done in the previous sprint and to assign tasks for the next sprint. This will be done in a similar manner; every two weeks there will be a meeting between the advisor and the author of this dissertation to discuss what has been previously done and what to do in the next two weeks.

During this first semester, the scrum methodology will not be applied. To have organization and estimation of the tasks to be completed, a Gantt chart was made. By doing this, we can have an idea on what to do, if we are on schedule and get a better feeling for delays when something goes wrong. It can be seen below in figure 1.1.

First Semester Gantt Chart Estimation

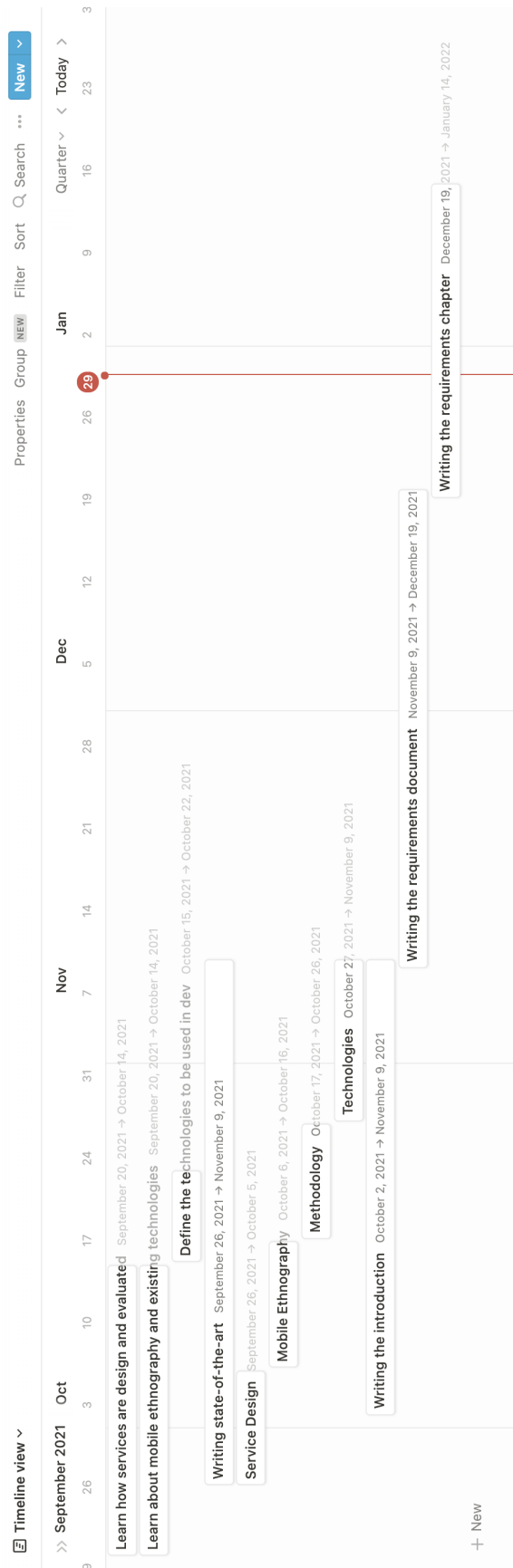


Figure 1.1: First Semester Gantt Chart Estimation

Figure 1.2 shows the real gantt chart. This chart accounts for delays and we can clearly see in comparison to the one done in beginning that some dates suffered some changes. This is completely normal, as not everything we estimate will work accordingly.

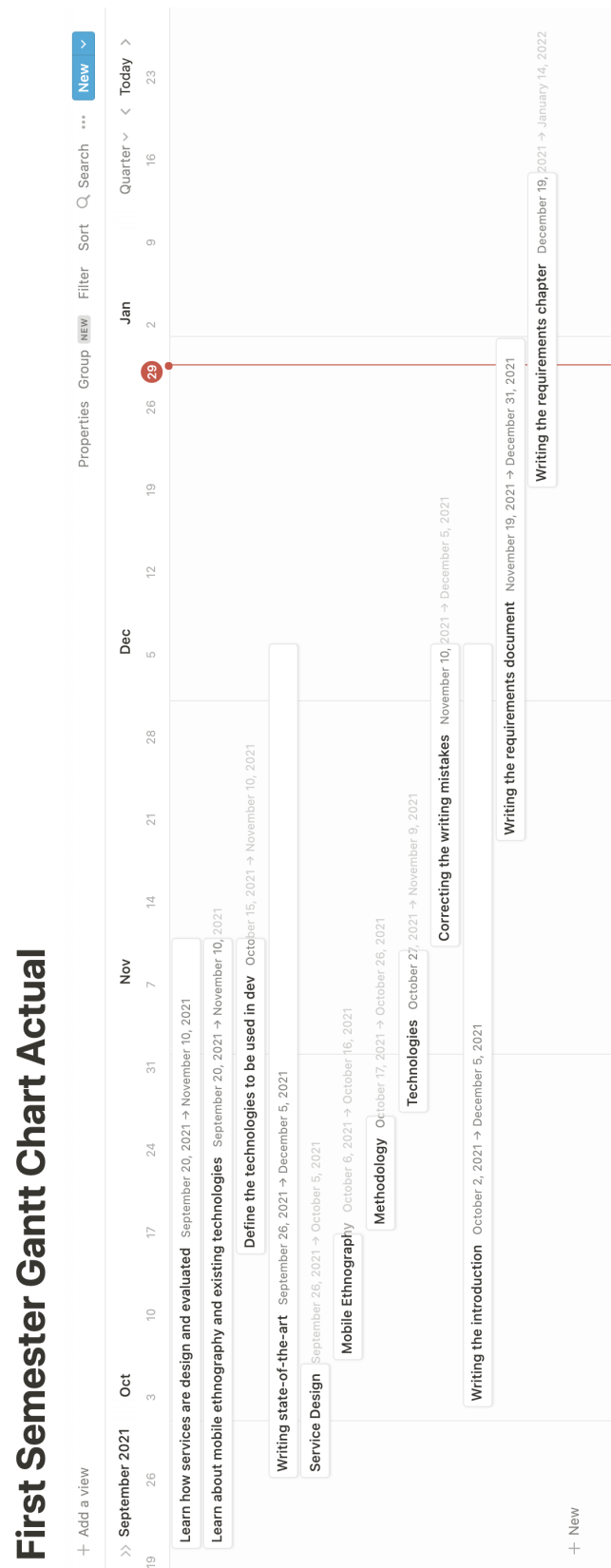


Figure 1.2: Actual First Semester Gantt Chart

For the second semester an estimation was also made and we can see it below in figure 1.3.

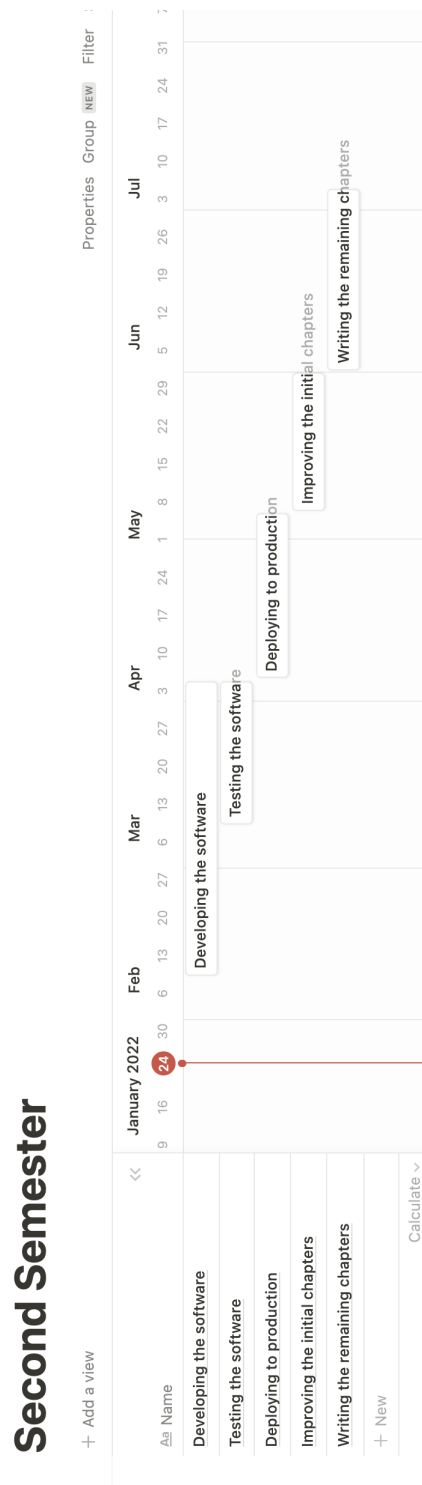


Figure 1.3: Second Semester Gantt Chart Estimation

In Figure 1.4, we can see the actual Gantt chart of the work done for the second semester. The difference from the estimation is quite significant. The divisions also include the time required for the chapter to be written. So, the division of writing the remaining chapters was removed. We can also see that the time for the development part to be completed was longer and the building of the software architecture was also added. This delayed software testing and, consequently, deploying to production was also postponed, despite taking much less time than anticipated.

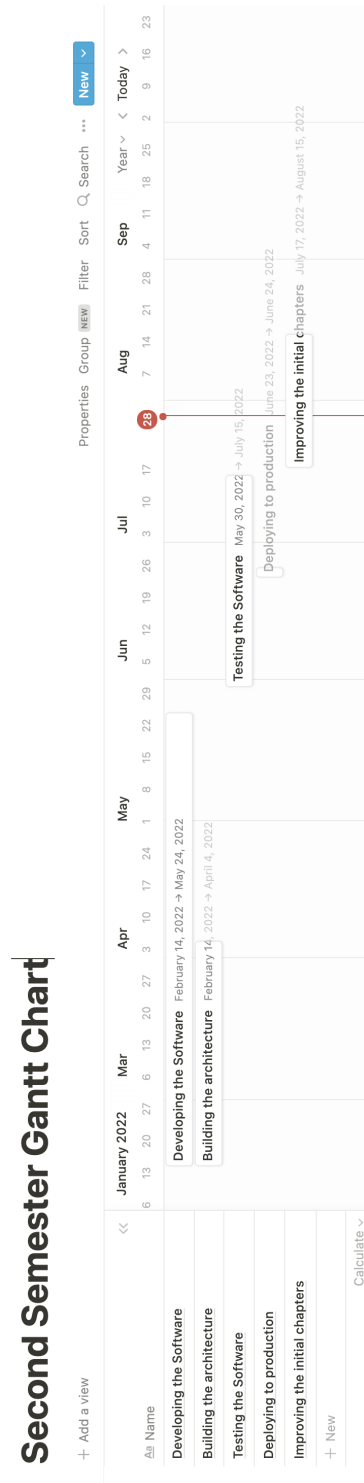


Figure 1.4: Actual Second Semester Gantt Chart

1.4 Report Structure

The rest of the document is organised in the following chapters:

- **State-of-the-art (Chapter 2)** - This chapter will display the knowledge obtained on the topics needed to produce the software tool in question.
- **Requirements (Chapter 3)** - This chapter will contain the requirements for the software tool to develop.
- **Software Architecture (Chapter 4)** - This chapter will contain the architecture, done with the C4 model, of the software tool to develop.
- **Software Development (Chapter 5)** - This chapter will detail the development stage of this tool.
- **Software Testing (Chapter 6)** - This chapter will explain and show the tests done after the tool is developed.
- **Conclusion (Chapter 7)** - This chapter will detail the whole project and will have the main conclusions about the project.

Furthermore, some annexes were also built:

- **Software Requirements Specification** - This annex contains the full specification of requirements to enrich the respective chapter, Requirements.
- **Software Installation** - This annex contains a guide for the installation of the software tool.

Chapter 2

State-Of-The-Art

In this chapter, the topics needed to produce the software tool will be presented and we will get a deeper knowledge about them.

The first section includes the procedure. Here, we can see the process on how to reach the topics for the rest of the state-of-the-art. The second section includes an explanation of service design. The third section shows what is mobile ethnography. The fourth section includes similar software tools to the one to develop and the last section has the chosen technologies for the software tool to develop.

2.1 State-Of-The-Art Procedure

To do this state-of-the-art, several topics were searched online. The first thing in this work was to learn how services are designed and evaluated. By taking the course Service Engineering, it helped to faster learn and comprehend how services work and how they are designed. After having this knowledge, a service was designed and built to better internalize the concepts learnt in the course. The service was a car dealership website that delivered the car to the customer's door. The technologies used in this service were the React framework for the front-end and Amazon Web Services (AWS) for the back-end. Mobile ethnography was also mentioned in this course, allowing us to gain knowledge that can be applied in this chapter.

The bibliography for Service Engineering was also useful, including the two books, *This is Service Design Thinking* [5] and *This is Service Design Doing* [4]. With these books, we can gain a deeper understanding of service design itself.

To complement the definition of mobile ethnography, there was a broader search for it. One of the first steps was doing google searches with the keywords *Mobile Ethnography* to have a better the definition of the topic. In addition, *ResearchGate* and *Google Scholar* were also used to find articles related to this topic.

Following this, there was a search for platforms similar to the one to be developed. One of them, *ExperienceFellow*, was introduced by the advisor of this dissertation, Prof. Paulo Rupino da Cunha. It was an example given in the Service Engineering course to teach concepts about service design and mobile ethnography. To better explain it by the author of this dissertation, a guide, Annex 1, that is on their website for people to learn how to use the platform was used. Another software tool, *Indeemo*, was found by searching for Mobile Ethnography in *Youtube* and also helped to better understand about Mobile Ethnography,

due to their videos explaining the theme. In an attempt to explain this platform and enumerate their functionalities, a contact with them was made by email, where they indicated a link on their website with an overview. However, this was very superficial and did not explain the functionalities in detail. They did not offer a free trial or a demo.

ExperienceFellow is a very good tool and by having a concise explanation of it will help in the next chapter to find the requirements and get concrete ideas on what to develop. More platforms were found by doing a google search with the keywords *Mobile Ethnography Platforms*. In result to this, the website [1] was found. However, they were just for very specific cases, like *QualSights*, which "is a platform for remote video observation, interviews and focus groups." [1].

2.2 Service Design

As stated in the Introduction (Chapter 1), services represent 65% of Gross Domestic Product (GDP) and most of the employment today in developed countries [6]. Everyone uses services and the number of available services is always increasing. For example, *Netflix* is a service that is highly successful. They went from 400,000 subscribers in 2001 to 209.18 million in 2021 [12]. We can clearly see that the need for services is high in demand.

Service design has not a clear definition and it is evolving over the time [5]. Everyone has different opinions about this topic. We can see this in the book *This is Service Design Thinking* in the following sentence, "If you would ask ten people what service design is, you would end up with eleven different answers – at least." (Richard Buchanan, 2001, [5])

As there is no concrete definition to service design, according to Mark Stickdorn in the book *This is Service Design Thinking*, there are 5 principles of service design thinking that can be applied [5]:

1. "**User-Centered** - Services should be experienced through the customer's eyes."
2. "**Co-creative** - All stakeholders should be included in the service design process."
3. "**Sequencing** - The service should be visualised as a sequence of interrelated actions."
4. "**Evidencing** - Intangible services should be visualised in terms of physical artefacts."
5. "**Holistic** - The entire environment of a service should be considered."

As "Services should be experienced through the customer's eyes" [5], user experience which "is how a user interacts with and experiences a product, system or service" [50], also impacts the customer satisfaction and consequently the service provider itself. It is important to have tools that can measure that.

Traditionally, service providers would use questionnaires to measure customer satisfaction, and we have an example of this in a case study referring to a Walking Quiz, which is a questionnaire, in the article *Thinking and Doing Ethnography in Service Design* [39]. According to *Oxford Reference*, Ethnography is "The scientific study of customs, habits, and behavior of specified groups of people, usually applied to tribes or clans of people in nonliterate societies." [36]. We can clearly see that ethnography does not only apply to service design. However, in our dissertation, we are only going to think about this topic

regarding service design. In our specific case, the main goal is to use mobile ethnography to collect data and, subsequently, to use service design to help service providers analyse the collected data. The software tool to develop aims to help improve services; thus, the data collected will be regarding services and how users interact with them. It will be analysed by providing tools that follow the five principles of Service Design [5] stated earlier in this section.

2.3 Mobile Ethnography

Considering the Service Design chapter, it can be said that there are more modern solutions to classical ethnography. We live in a world where almost everyone in developed countries has a smartphone [47]. Due to this, the better solution in need is Mobile Ethnography. Currently, this is not being developed in the way that is needed, there are too few platforms in a world dominated by services. If a search is made with the keywords *Mobile Ethnography Platforms*, we can find some platforms. However, they do not have the popularity or users needed to make the service design overall better.

Mobile Ethnography is a qualitative research method that relies on using mobile devices for data collection, and therefore, takes advantage of technology to document, analyse and derive implications of real-time customer experience [32]. Participants in a project self-document their experiences in real-time and researchers analyse this data. Researchers can reach participants by sending notifications as a reminder to collect data or for some kind of guidance [4].

With mobile ethnography, we can have a larger number of participants in a study, "A mobile ethnography project might include 10, 100, or even 1,000 participants documenting their experiences with a brand, product, service, event, or similar." [4]. Also, "Mobile ethnography works well for longer research over one or a few days, as well as for rather intimate subject matters people hesitate to talk about with others." [4]. This is good for sensible topics where a study has to be made or when there is a need for collecting large amounts of data of each participant.

Mobile Ethnography can have several applications. There are some cases, such as in tourism, health, and retail research, in which mobile ethnography is applied to do some studies shown in this article [32].

As the participants self-document their experiences, this has to be analysed later. A researcher uses a computer for that purpose. We can see that in the studies done in this article [32], the researchers use computer aided analysis systems, like *NVivo*. These computer aided analysis systems are only to analyse the data and they do not do the collection. However, our goal is to create a software tool that can do the collection and analysis and be all integrated in one place. This allows the researcher to not having to waste time in exporting the data to use some other software to analyse the data.

2.4 Mobile Ethnography Software Tools

The two software tools found in a service design context, *Indeemo* and *ExperienceFellow*, are tools for evaluating customer experience using mobile ethnography. However, *ExperienceFellow* was discontinued in March 2021. There are other tools that use Mobile Ethnography, but they are for different usages rather than Customer Experience Evaluation of services. We are going to talk about the *ExperienceFellow* tool in this section and its functionalities to have some context to do the requirements of the software tool to develop. We also wanted to explain the *Indeemo* platform, but this was not possible because the platform is paid and there is no free trial or demo available.

In this section, we focus on the *ExperienceFellow* platform. This software consists of having a website for researchers to analyse data and a mobile application for participants to capture data.

ExperienceFellow Mobile Application

The mobile application is like a diary on the participant's mobile phone, they give insights of their experiences to the researchers. Whenever they want, they can add an experience.

When the participant enters the application, he can sign up or sign in. When signed in, the participant reads the QR code to enter a project.

Imagine that we are entering a hotel and this hotel uses *ExperienceFellow* to get some insights to improve their customer satisfaction. The hotel can display their QR code, for example, at the reception, and when the participant checks in, he or she is informed about this application, and after downloading and signing up, he can scan the QR code and send valuable information to the hotel. This uses mobile ethnography, as hotel users will provide their real-time experience in an ethnographic way. We can see in the next paragraphs how the participant gives the information and that this is a qualitative research by the data provided. Also, all data is user-centered, this means that it is related to a user. Mobile ethnography follows an user-centered approach on the data collection [48]. At the outset, it follows one of the principles of service design, despite the fact that the data has not yet been analysed.

Every time the participant wants to give their opinion about something, for example, the minibar was empty when he first entered the room, they can enter the app, choose that project and then add a touchpoint. This touchpoint consists of having some parameters:

- **Title** - It is a name for the touchpoint, in this case, just a brief name.
- **Emotion** - Rating from 1 to 5 with smiley faces, 1 being the worse and 5 being the best.
- **Notes** - In this field, the participant can detail what exactly happened.
- **Media** - Can add photos or videos of the situation.
- **Location** - The participant can choose to send his GPS coordinates or not.

Also, in the touchpoint, a timestamp is created automatically to know when that situation occurred. Returning to the example of the empty minibar, we can illustrate how it works with a real example:

- **Title** - Minibar is empty upon check-in.
- **Emotion** - 2
- **Notes** - When entering the room, the minibar was empty. This is unacceptable as it is a very expensive hotel.
- **Media** - The user would upload a photography of the empty minibar.
- **Location** - The user can choose to send his GPS coordinates or not.

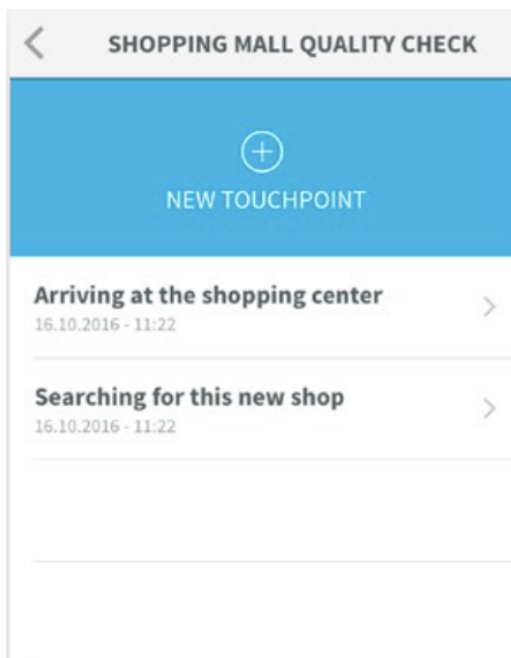


Figure 2.1: Inside the project page in the Mobile Application. Figure taken from Annex 1.

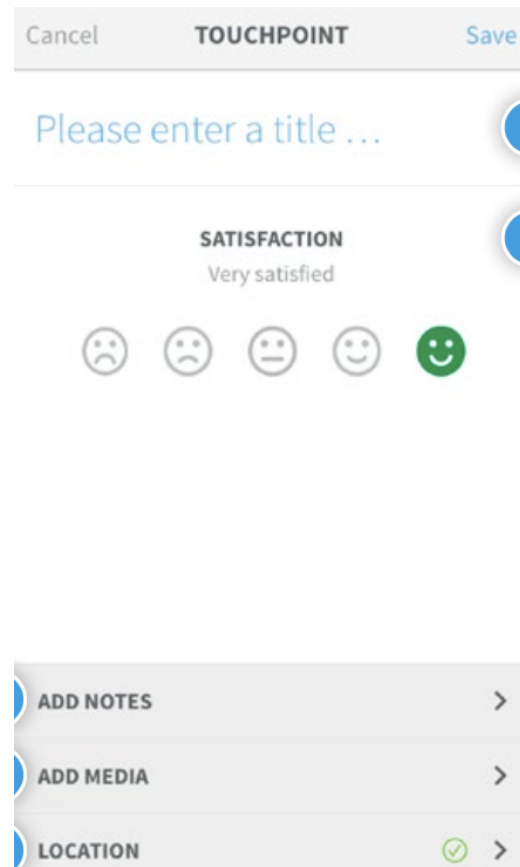


Figure 2.2: Inside the create new touchpoint page in the Mobile Application. Figure taken from Annex 1.

In Figure 2.1, we can see the page in the mobile application where we can add a new touchpoint to a specific project by clicking the *new touchpoint* button. In this case, the project is *Shopping Mall Quality Check* and we can also see the touchpoints already submitted, *Arriving at the shopping center* and *Searching for this new shop*. In Figure 2.2, we can see the page in the mobile application where we insert the data from the new touchpoint that we want to save.

Another feature of the application that we have not talked about yet is deferred syncing, which is the ability to use the app offline. For example, the participant could not have internet in the hotel, however, he can add the touchpoint that the minibar was empty and later when a internet connection is available, this information would be uploaded. Also, the participant can receive push notifications from the hotel, this is a way of reminding them to use the application.

ExperienceFellow Website

On the website, the researcher or the service provider can sign up or sign in. After signed in, he can create a project or choose an existing project. Each project is divided in four stages:

1. Project successfully created and set up;
2. Collecting data;
3. Stop collecting data;
4. Archive Project;

Inside the project, the researcher or service provider has a page to invite participants, here, each project has a QR code for the participants to enter using the mobile application, this could be disclosed by email or it could be printed to have it in a paper format.

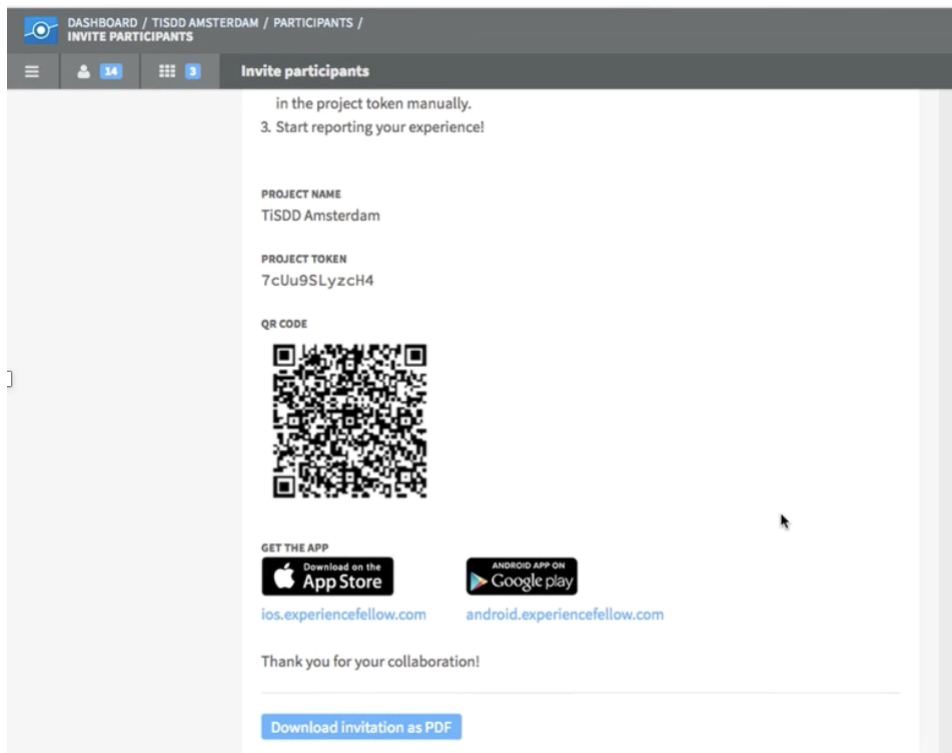


Figure 2.3: Example of the QR code to invite participants

In Figure 2.3, we can see the project name, the token for the created project and the respective QR code. On this page, the researcher or service provider can also download the invitation as a PDF in the blue button.

There is a functionality inside of each project that is the raw data. This data is shown by having a line of each step that a participant took, the step here is a touchpoint, it shows the title, the timestamp and the emotion of the touchpoint. This is shown step-by-step and in order so that we can visualise the journey of each participant. We can also see the emotions, what is positive and what is negative. Then you can click on top of a touchpoint and a pop-up will appear with all detailed information.

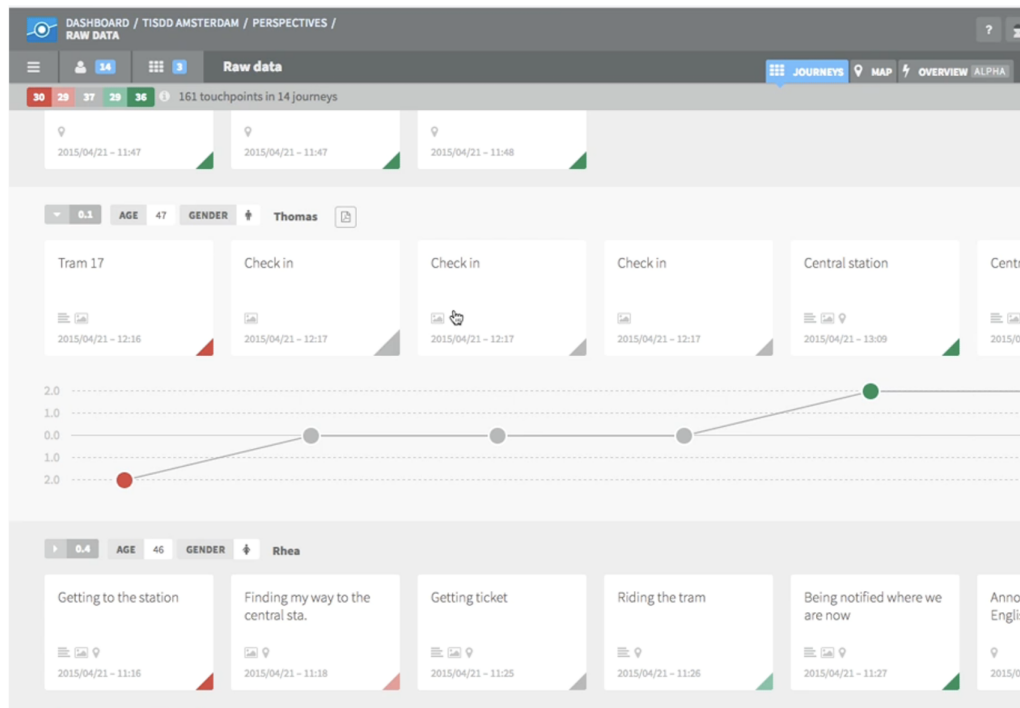


Figure 2.4: Example of the Raw Data functionality.

In Figure 2.4, we can see the raw data functionality, we can see that each line shows the data for a specific participant. Each participant has his journey, and we can visualise the several touchpoints. There is a color for each touchpoint, red indicating a worse rating and green a better rating. Also, the gender and age are also shown.

There is the perspective functionality as well, here the data can be changed. This page is similar to the raw data page. However, here, the data can be changed by dragging the touchpoints to the place that the researcher wants, and blank spaces in the middle of the touchpoints can be added. The main objective in this page is to compare different participants and their journeys. You can also add tags to each touchpoint to later on apply filters on it. The filter can be applied to text, tags, emotions, name, gender or age. This will only show the touchpoints that can relate to that filter. You can also combine filters. This will make it easier to find patterns in the data set.

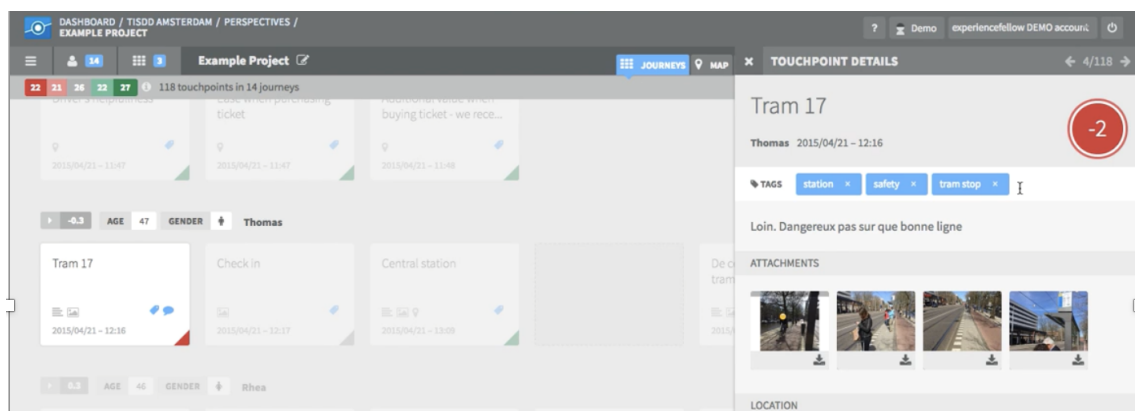


Figure 2.5: Example of the Raw Data functionality.

In Figure 2.5, we can see that by clicking on the top of a touchpoint, we can add filters and see a more detailed view of it. In this figure, the filters applied were station, safety

and tram stop.

There is also a functionality called the overview. This splits up the data, and now we enter into a mix of qualitative and quantitative research. In the overview, you can filter by different tags and then we can split the data by emotion, we can see what the most negative and the most positive touchpoints are. The touchpoints are represented in small squares of colour, and if we click on one, it will appear the pop-up of that touchpoint with all detailed information.

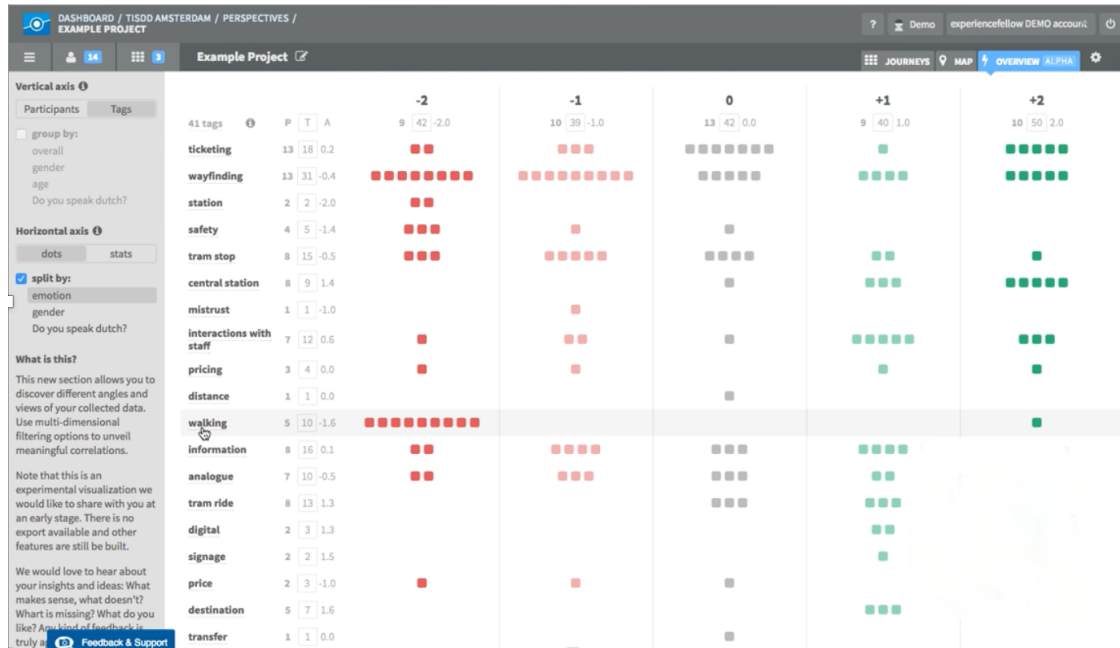


Figure 2.6: Example of the overview functionality.

In Figure 2.6, we can see the overview functionality. This specific case contains the tags on the vertical axis and the emotions on the horizontal axis. In addition, the researcher can choose the data he wants in the horizontal and vertical axis.

Another functionality is a map that shows all the touchpoints and we can see the clusters of good and bad emotions. In this map, each pinpoint has a colour and in the middle of the pinpoint it shows the colour of their emotions, from red being the worst and green being the best.

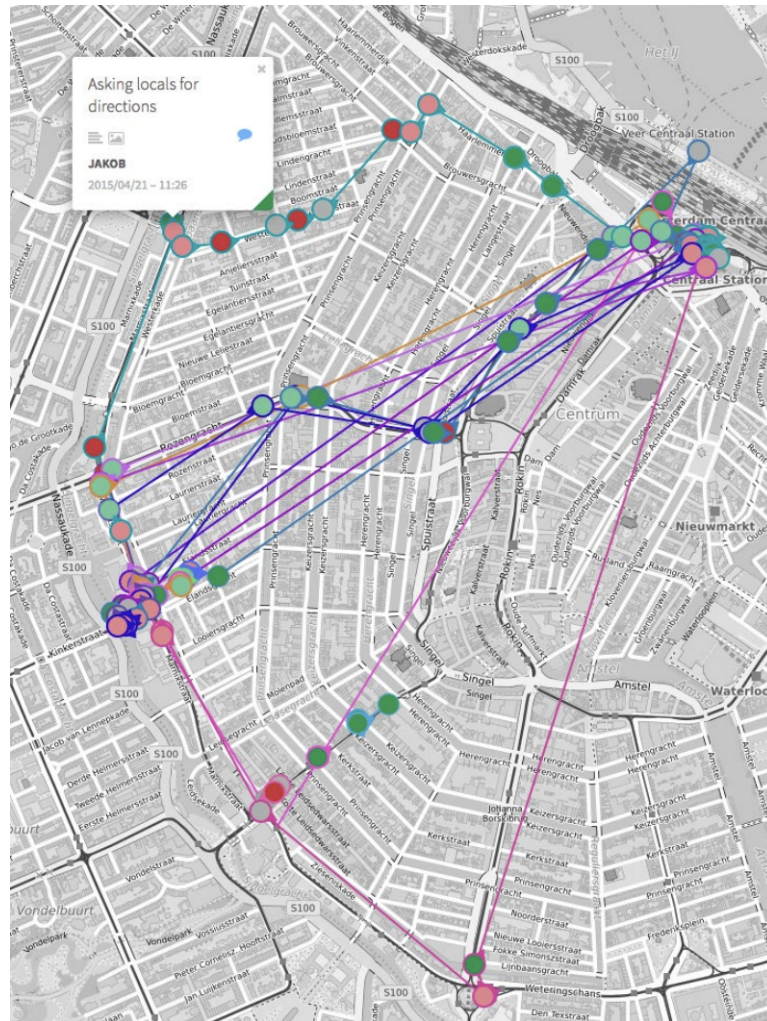


Figure 2.7: Example of the map with the touchpoints

Regarding the functionality to export data, this can be done in three different formats:

1. PDF;
2. Excel;
3. ZIP file that includes the excel file and all photos and videos;

2.5 Technologies to help build the Software Tool

In this section, there will be a study of technologies to build the software tool. We are going to decide all technical aspects for development of the tool, mainly, the front-end for both the website and for the mobile application and the back-end. There will be also a study to decide the hosting of our tool.

2.5.1 Website Front-end

Using Javascript frameworks allows to build highly visual and reactive websites with beautiful designs. According to a stackoverflow survey, which is a platform programmers use to clarify their doubts, the most popular language is Javascript with a percentage of 69.7% in 2020. In relation to frameworks, there are a lot popular frameworks for building websites, the most popular according to the survey from stackoverflow is jQuery, followed by React.js and then Angular [43].

There are a lot of different Javascript frameworks. In the following sections we are going to compare some of the most popular that are used for web development. The ones that we are going to compare are jQuery, React.js and Angular.

jQuery

According to their official website, "jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript." [21].

This javascript library is the most popular due to being used in WordPress which is a low-code platform to build websites [13].

By using a technology like this we are able to manipulate the Document Object Model (DOM) in a simpler manner, making a more responsive website. The documentation for this framework is good and very easy to understand. By being the most popular framework in stackoverflow, the support is also very good.

React.js

According to React's Wikipedia page, "React.js is a free and open-source front-end JavaScript library for building user interfaces or User Interface (UI) components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality." [49].

In terms of libraries, React.js has an abundance of them, they can help the developer to create a more visual and interactive UI and it also helps for the website routing, for example, the use of the libraries HashRouter [38] and BrowserRouter [37] that allows for the user to navigate in the website. Also, there is a library called Redux [19] that allows for better state management and with that we can spend less resources from our servers

and allow for faster speeds of the website in development. In terms of documentation it is outstanding, as their official website shows a very visual representation of the code and it allows to learn this framework very fast in terms of state management for a person already familiarized with javascript [18]. In terms of support, this framework also shows very good support as it the second most popular in stackoverflow and the bugs are being fixed everyday as it is an open-source project and it belongs to one the most popular companies in the world, Facebook. If we search through their github [16], we can see the React.js repository and can see their fixes.

This framework is already very well known by the author of this dissertation, as it was used in several previous projects. The possibility of using this in the development is very high for the at ease that the author has with this framework. However, this framework is mainly for the front-end, but there are Software Development Kit (SDK) that can help the process of developing it all in React.

Another feature is that in React you use JSX which is a syntax extension to Javascript. It is very straight-forward. "JSX can remind you of a template language, however JSX comes with the power of Javascript." [17].

Angular

According to Angular's official website, "Angular is a framework for building interfaces of application using HTML, CSS and mainly Javascript. Angular is a development platform, built on TypeScript which is a more robust Javascript built by Microsoft. As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code"

Like React.js, Angular also relies on components. "Components are the building blocks that compose an application. Every component as an HTML template that (...)", like React.js, "(...) lets you alter the rendered DOM using states. Another feature is Dependency Injection, this lets you declare the dependencies of your TypeScript classes without taking care of their instantiation. Instead, Angular handles the instantiation for you. This design pattern lets you write more testable and flexible code. Even though understanding dependency injection is not critical to start using Angular, we strongly recommend it as a best practice and many aspects of Angular take advantage of it to some degree. There are also libraries, some of them are first-party libraries, in other words, they are the essential libraries to build web applications in Angular, they include:

- **Angular Router** - Advanced client-side navigation and routing based on Angular components. Supports lazy-loading, nested routes, custom path matching, and more.
- **Angular Forms** - Uniform system for form participation and validation.
- **Angular HttpClient** - Robust HTTP client that can power more advanced client-server communication.
- **Angular Animations** - Rich system for driving animations based on application state.

- **Angular PWA** - Tools for building Progressive Web Applications (PWAs) including a service worker and Web app manifest.
- **Angular Schematics** - Automated scaffolding, refactoring, and update tools that simplify development at large scale."

[28]

Appropriate Technology for Website Front-end

Having all the topics of these frameworks in mind, the technology chosen for the front-end will be React.js. The author already knows and is proficient in this framework, this is a plus because the author does not have to learn other frameworks. In addition, React.js is the framework that has the best documentation and libraries to create a highly visual and responsive website. Being open source and built by Facebook, it has optimisations such as using the library *Redux* [19], which can help reduce resources costs, for example, the number of reads from the database.

We are going to build our platform based on the *ExperienceFellow* existing one. One of the functionalities of this platform requires drag and drop. This is very simple to do in React.js, we only have to use a library called react-beautiful-dnd [3]. Another feature of *ExperienceFellow* is the touchpoint overview, which uses a map to see the clusters of good and bad satisfaction ratings. In React.js there is a library called google-map-react [27], which allows us to have access to Google Maps just by having an API Key from Google. This is also a plus for our website.

2.5.2 Mobile Application Front-end

Regarding the mobile application, we have two choices, a native one or a Progressive Web Application (PWA). We are going to opt for doing a PWA. The reason for choosing a PWA over native is the reduction in deployment cost. With a native application, we need to pay a certain amount to Google's Play Store and Apple App Store. We do not have those resources available. However, by having a PWA we only need a server and a domain to deploy the application, and Department of Informatics Engineering (DEI) provides these. One of the disadvantages of a PWA is that is slower to respond than a native application.

This application can be built by using a cross-platform framework. This allows to code in Javascript for example, and then it is rendered into a PWA. This makes development and testing easier because it is fast to build and deploy the PWA.

Today, there are many cross-platform frameworks that exist. However, if we are going to use React.js for the website front-end, we could also use React Native for the Mobile Application. The semantics of the two frameworks are very similar.

Based on the *ExperienceFellow* application, we need to be able to access the camera, the location and receive push notifications.

React Native has very good features that can help create a mobile application very fast. Like React.js, there is a huge amount of libraries that can create highly visual and interactive mobile applications. Also there is a SDK, Expo [14], that has features that are essential in our Mobile Application. These include:

- **Camera-** This allows to use the back camera to take photos or videos and to read QR Codes.
- **Location** - This allows to use the geolocation information from the device.
- **Notifications** - This allows to receive push notifications.

This SDK simplifies building our application and also simplifies the testing for the application. There is a mobile application that can instantly display the software that we are developing in real time on the mobile device. When the code is compiled, a QR code is shown, and inside the application, we point the camera at it. This allows us to test on different devices in a quick way.

React Native is built by Facebook and it is one of the most popular frameworks to build cross platform mobile applications [44]. This means that in terms of support and documentation, like React.js, it is very good. The only other platform that has similar popularity is Flutter. However, it does not allow us to build such beautiful front-ends as it does not have so many libraries.

By using Expo, we can build our application into a PWA or into a native application for both iOS and Android. There are only small changes that need to be made in our code for transforming from a native application to a PWA. These changes are related to the libraries that we are going to use, as not all work in the PWA.

2.5.3 Back-end

According to data from statista [45], nowadays the cloud computing is the big thing, due to the smart approach of pay-as-you-go, which allows the owner of the website to only pay for the resources used. Cloud computing is growing at an exponential level [45]. There are several services in existence. The two most popular are AWS and Firebase from Google.

AWS

AWS is the cloud service from Amazon, they include a lot of features in their service. As the author already had worked with AWS, the main features of this application should have a database and a hosting feature.

For hosting and as the front-end is built in React.js, we could use Amazon Simple Storage Service (Amazon S3) for static hosting, and with React.js this is enough. Amazon S3 is offered by AWS and provides object storage through a web service interface. We could also use this for storage the pictures and videos of the participants of our project.

For using Amazon S3 we also need lambda functions to run code in the back-end. This is another service from AWS. This service lets you run code on the cloud, like serverless computing. However, we cannot do it without using another service to send the data from the front-end to the back-end. The service in question is Amazon API Gateway. It allows to create RESTful Application Programming Interface (API) and WebSocket API. This will allow you to directly send data from the front-end to a lambda function and get a response. Regarding the lambda functions, we can write and read from the databases and from Amazon S3.

In AWS we have several choices for the database. They are divided by categories that include

- **Relational databases** - Amazon Relational Database Service (Amazon RDS) and Amazon Aurora
- **Key-value and document data** - Amazon DynamoDB
- **Graph databases** - Amazon Neptune
- **In-memory databases** - Amazon ElastiCache
- **Search** - Amazon Elasticsearch Service

After having some databases, we have to choose which is best for our project. In this case, there would be a key-value and document database like Amazon DynamoDB that allows us to divide the data by several people and by several projects that exist on our platform. Amazon DynamoDB has high-throughput, low-latency reads and writes and an endless scalability. This is good if we think in terms of having a lot of participants and a lot of researchers manipulating data.

Finally we can do a cost estimation using AWS calculator [2]. For this, we are going to suppose a big number of requests from the API, 100000, a considerate amount of data stored in the database, 1GB, and also 20GB of data stored in the Amazon S3. This would give us a total amount of roughly 50 dollars per month. As we do not have any money allocated for this project, AWS is not viable for our project.

Firestore

Firestore is a platform by Google to use in mobile and web applications. The features that their platform that are interesting for the software tool that we are developing are:

- **Firestore Hosting** - "Firestore Hosting provides fast and secure hosting for your web app, static and dynamic content, and microservices." [25].
- **Firestore Database** - "It is a flexible, scalable NoSQL cloud database for mobile, web, and server development from Firestore and Google Cloud." [23]. Also, it allows for deferred syncing, which will be a requirement of our Mobile Application.
- **Authentication** - "Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices. Firestore Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more." [22].
- **Cloud Storage** - "Cloud Storage for Firestore is a powerful, simple, and cost-effective object storage service built for Google scale. The Firestore SDKs for Cloud Storage add Google security to file uploads and downloads for your Firestore apps, regardless of network quality." [26].
- **Cloud Functions** - "Cloud Functions for Firestore is a serverless framework that lets you automatically run backend code in response to events triggered by Firestore features and HTTPS requests. Your JavaScript or TypeScript code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers." [24].

Firebase has a free tier, which can be enough for an internal project at DEI. In terms of authentication and registration, we can have an unlimited amount of users. For cloud firestore, there is a total amount of 1GB for the database, 50000 reads per day, and 20000 writes per day. For cloud storage, there is a total amount of 5GB, to surpass this limitation, image compression can be done to reduce the amount of space spent. If the storage exceeds 5GB, each extra GB is 0.0026\$, which is a very reduced amount.

With the available SDK for React.js and React Native, Firebase is a very good solution to our project as it can less time to integrate some features such as, deferred syncing and it is very simple to do user authentication. Also, the need for middleware, for example an API using Express.js, is almost none.

PostgreSQL

PostgreSQL is a "open source object-relational database". One of the main advantages of this database is that is open source, this means that are no costs associated with it. The only cost would be the hosting platform. However, DEI has their own cloud servers. One of the drawbacks of this application is difficulty to set up. An API has to be created. For creating it, we could use Express.js, which uses Node.js as the language. Another drawback is the difficulty of doing deferred syncing, we would have to do some more coding to reach the same level of Firebase.

Also, functions like sign-up and sign-in have to be coded, and user data has to be encrypted in the database. Firebase already does all of this automatically. Another feature that is not available is the storage, we would have to store every image in the server and this is complicated to set up in comparison to firebase, as firebase already gives a download link when we upload a photography or video to the storage.

Appropriate Technology for Back-end

Looking for the stated technologies, the better option in the author's opinion is Firebase, he also has used it in the past and has features like deferred syncing that are already implemented when the SDK is used. This is optimal for our mobile application. Also, Cloud Firestore is very scalable as a database, which can be good for us when the software to develop reaches a large number of users. Also, it has a free-tier that AWS does not have, which can be good for developing before putting our software tool to production.

As said in the PostgreSQL section, sign-up, sign-in and storage are simpler to do in Firebase and makes the database GDPR compliant due to the automatic encryption of user data.

2.5.4 Hosting

For hosting our website and PWA, we can also rely on DEI Cloud. This is service in from the department that includes a Virtual Private Server (VPS) with limited quota. Each machine has 4GB of RAM, 4 Virtual CPUs and 100GB of storage. This is good for our project as it does not have any costs. The downside of using this technology is the difficulty to set up the server.

This page is intentionally left blank.

Chapter 3

Requirements

In this chapter, requirements for the software tool were collected. Several methods were used. For functional requirements, the methods used include a use case model, a user interface model and a MoSCoW prioritisation. Also, non-functional requirements were thought and the most appropriate for our software tool were chosen. In addition, a separate document was written. This document is an Annex with the name Software Requirements Specification and has the full requirements for this project. It contains the following chapters:

1. **Introduction** - In this chapter, the purpose of the application and the risk is stated.
2. **Overall Description** - This chapter shows a more detailed description of the product to be developed.
3. **Specific Requirements** - This chapter contains the Specific Requirements, starting with a Use Case Model that contains a Context Diagram, Use Cases Diagrams and Use Cases. Then, the User Interface Model was done, it includes User Experience Diagrams and the User Interface of the product. Finally, the Non Functional Requirements were stated and explained. To sum up this chapter, a list of all functional requirements was made with the MoSCoW prioritisation method applied to each one.

3.1 Requirements Gathering

The first step to reach the requirements for the software tool to developed was doing a study about the topics of this thesis, this includes mobile ethnography and service design. Subsequently, there was a search for similar tools on the market, and the functional requirements were taken from there.

With functional requirements in mind, there was a comprehensive thinking about the architecture and how everything is correlated. The research done in the state of the art served as a basis for this rationale. In Figure 3.1., we can see a sketch of how the software will be architected:

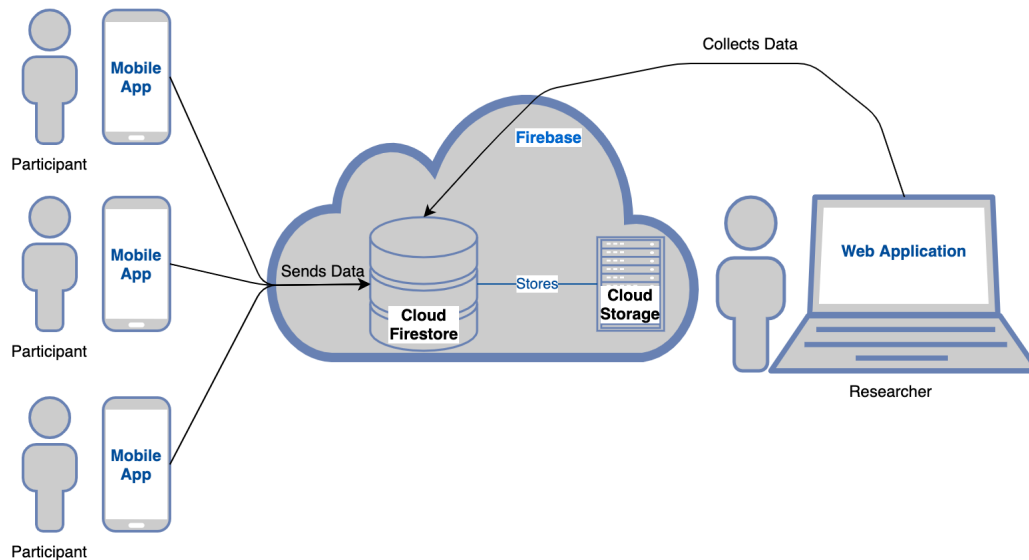


Figure 3.1: Architecture Sketch

The architecture shown here is just a rationale for how the software tool will work. Further ahead, a complete architecture will be made using the C4 Model. Afterwards, we went with more detail, separating it into two distinct areas, mobile and web, analysing the details of each area. To add to this, there was an analysis of users and possible constraints.

Since we already had all the necessary information about the software to develop, it was time to start on the specific requirements. For this, we started with a context diagram to identify the actors of our system. They are the participant and the researcher.

The next step was to advance to the requirements for each actor. For this several use cases diagrams were made. With this, we can state all requirements. After having all of them, we described each one with a use case.

The follow-up of having all the specific requirements is the design of the UI. The first step taken was developing an User Experience Diagram (UED) for the mobile application and for the web application. This will help us to develop our interfaces into a coherent design. By having an UED, the screens of the applications were easier to develop as we have all the information of each screen already stated. The design of the screens may not be final; however, they are an indication of what we are going to develop.

Finally, the non-functional requirements were discussed with the thesis advisor, and the most appropriate were chosen for our project.

3.2 Use Case Model

In this section, actors, use cases, and their relationships will be presented. We want to know how different users behave with regard to the system, with the purpose of solving any problem. The actors will have goals that can not always be fulfilled. We also have the interactions between the user and the system. Having this, we could know how to reach the goals. In the use cases diagrams, we can visually see what the goals are. Simplifying the final solution, which are the use cases themselves.

In Figure 3.2, the context diagram can be seen. It represents the interaction between the actors and the system. Our primary actors are the participant and the researcher and can

be seen on the left side of Figure 3.2. The system is displayed on the right side.

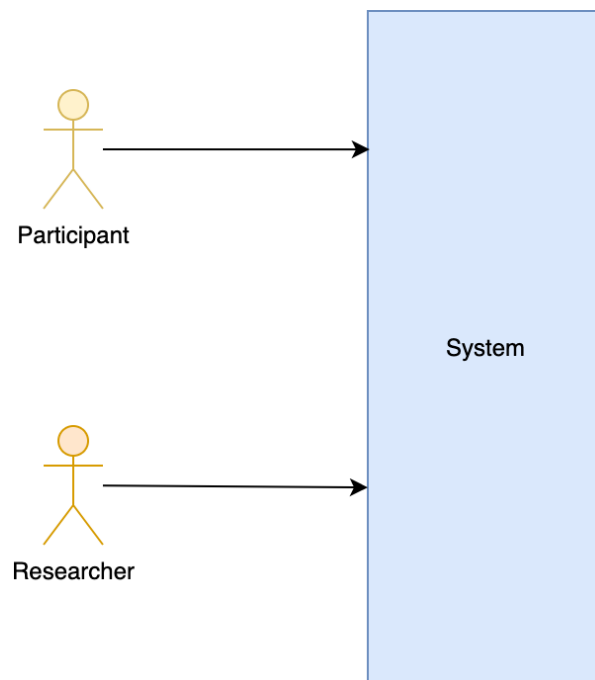


Figure 3.2: Context Diagram.

After having the context diagram, the use cases diagrams were made with the intention of showing the functionalities of our software with regard to each actor. In Figures 3.3, 3.4 and 3.5, we can see the functionalities that are related to each of the actors.

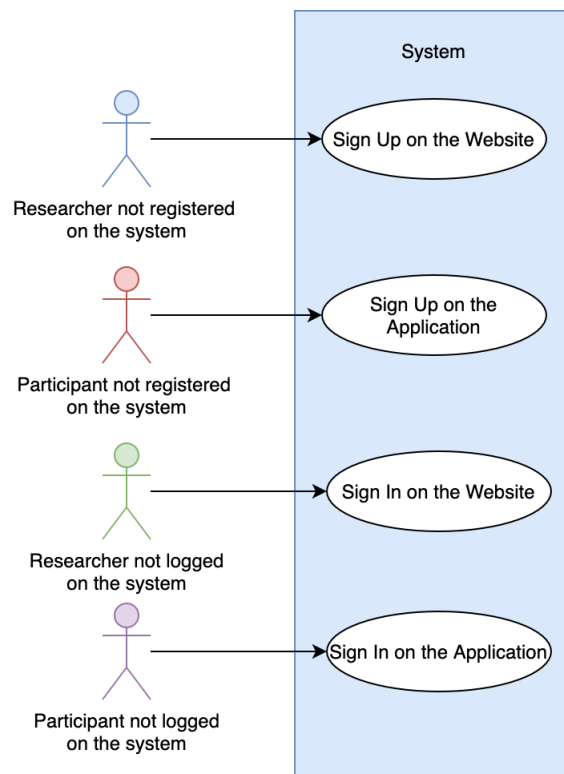


Figure 3.3: Use Cases Diagram for Signing In and Signing Up.

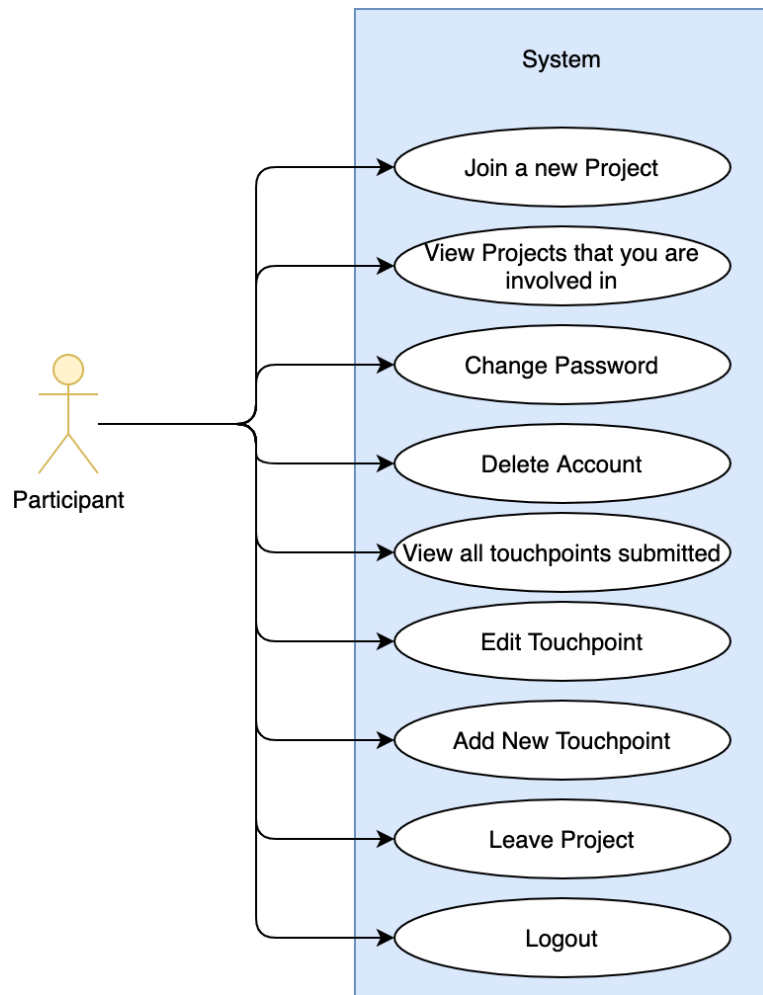


Figure 3.4: Use Cases Diagram for Participant Functionalities.

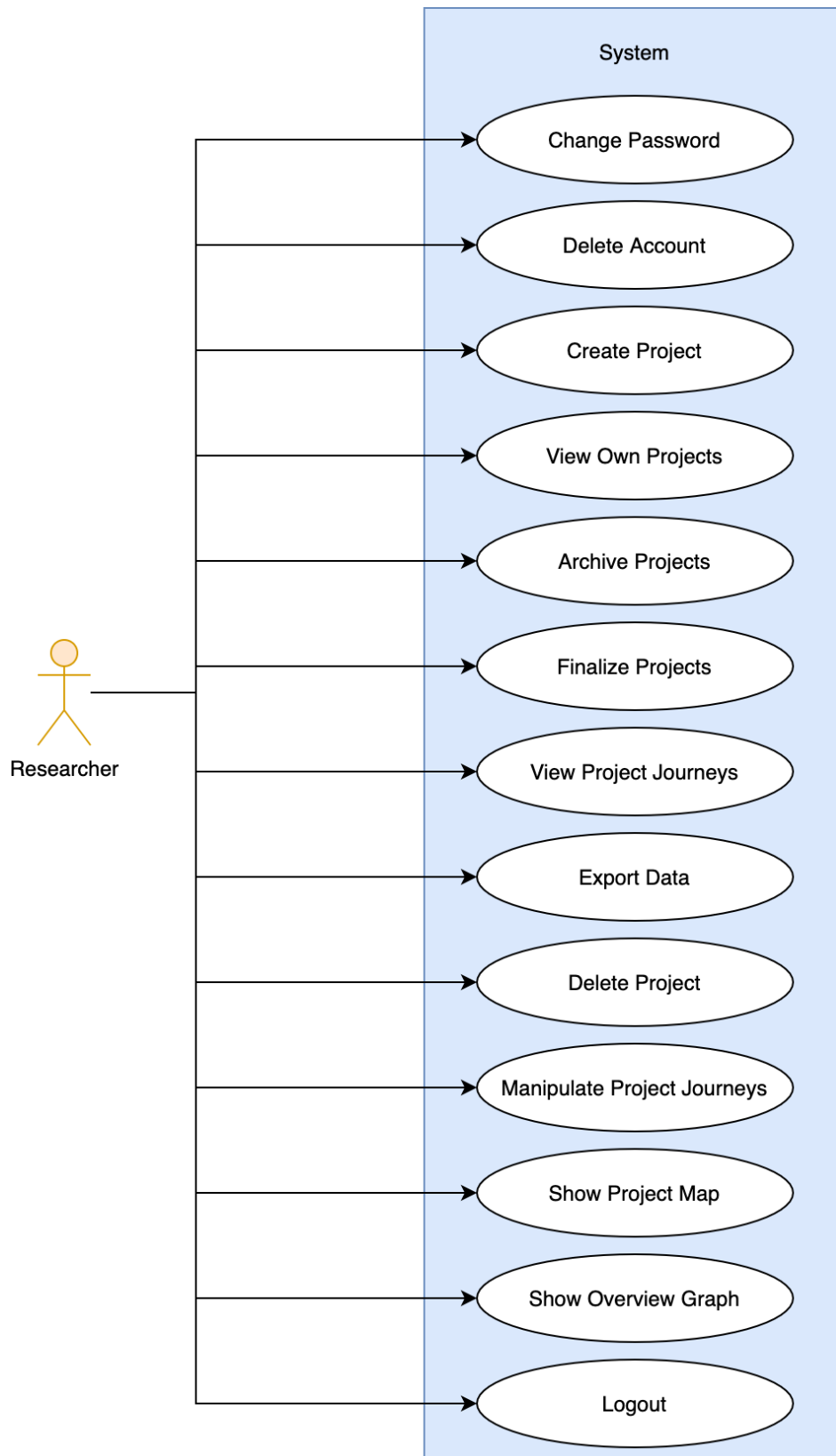


Figure 3.5: Use Cases Diagram for Researcher Functionalities.

With the diagrams made and the functionalities defined and related to each of the actors, the next step is doing the use cases for each functionality. In Table 3.1, an example of a use case can be seen. In this example, a new participant saw the application on the internet and he wants to sign up on it.

Name	Participant Sign Up
ID	01
Primary Actor	Participant not registered on the System
Level	Blue
Description	The participant makes an account on the system, inserting his personal data
Pre-condition	The participant is not registered and has the application installed on his device
Basic path	<ol style="list-style-type: none"> 1: The participant opens the application 2: The participant selects the Sign Up option 3: The participant Inserts the personal data 4: The participant accepts the legal terms and conditions 5: The Participant selects the Sign Up option 6: The System verifies the data 7: The System validates the data 8: The System adds the new participant
Alternative path	<ol style="list-style-type: none"> 6.a: The System finds an error in the participant data 6.b: The System informs the participant about the error (email already exists or password too weak) 7.a: The system notifies to change the personal data
Post-condition	Participant is registered and he is redirected to the Choose Project page
Frequency	Medium

Table 3.1: Participant Sign Up Use Case Table

The rest of the use cases can be seen in Appendix A - Use Cases or in the Annex with the name Software Requirements Specification. Each of the functionalities represented in the use cases diagrams is detailed as in Table 3.1.

3.3 User Interface Model

To aid in the development of the UI, the first step was to build a UED. This will help us to develop our interfaces into a coherent design. We can see in the UED the main focus areas, in other words, the several screens of our applications. We correlated the various focus areas to also understand navigation in our application. In each focus area, we can see how the user interacts with the application and also the types of data that are used. User Data is related to any type of data from the user of the application, email, and password in our case. Project Data is all of the data related to project in a broader sense, name of the project, token, etc. Touchpoint Data is the specific data for each project, and it is more detailed than Project Data. It includes the several touchpoints of each project.

In Figure 3.6 , the UED for the website can be seen. In this diagram, the several screens for the website can be seen and have the necessary information to help in the design. The connections between the screens are also displayed to help in the navigation between them.

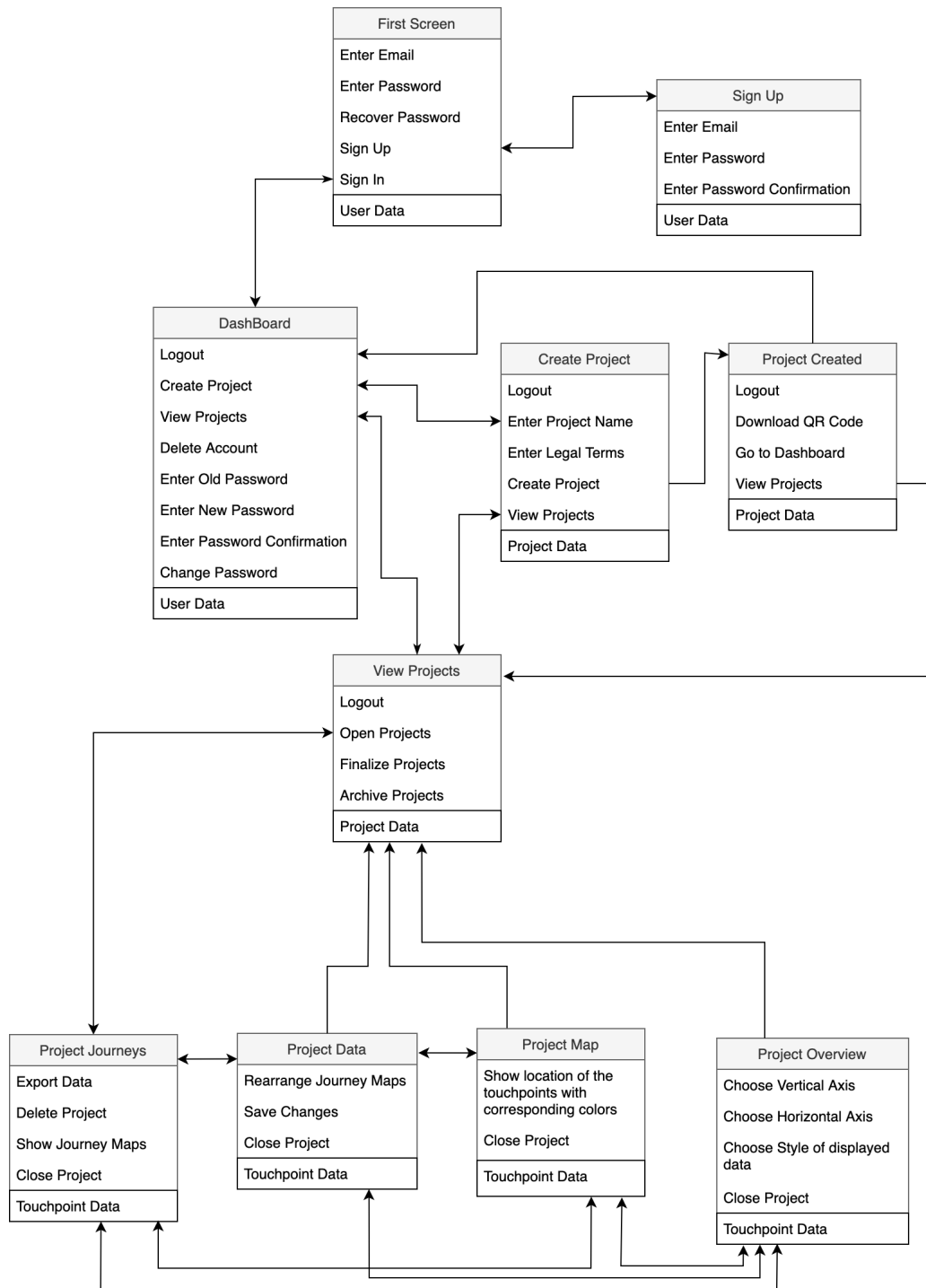


Figure 3.6: User Experience Diagram of Website

In Figure 3.7, the UED for the mobile application can be seen. It follows the same logic as the diagram in Figure 3.6. With these two diagrams we have the rationale to start creating the UI.

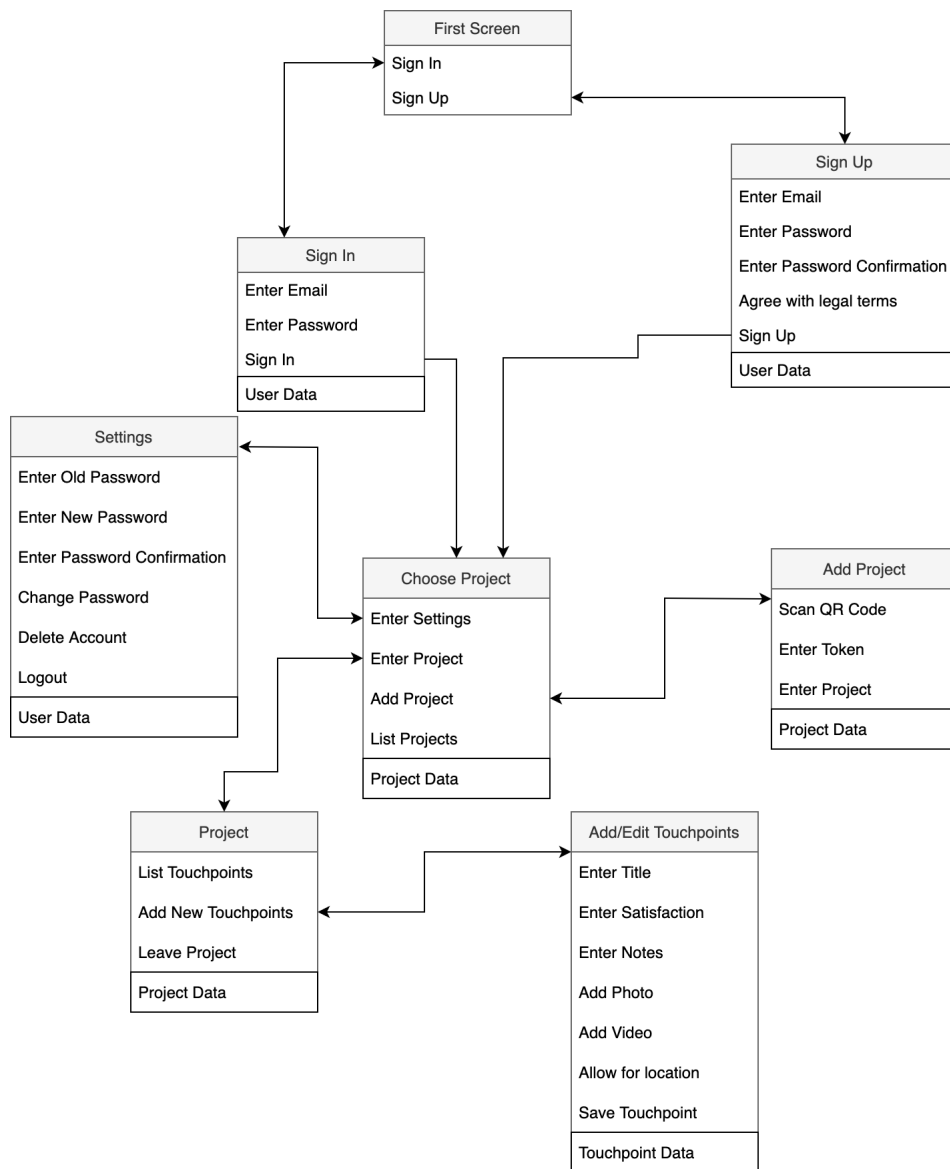


Figure 3.7: User Experience Diagram of Mobile Application

For the creation of the UI, a tool called inVision was used. This tool allowed to use preexisting libraries with already created modules such as login forms and buttons and allowed to drag and drop the modules and create the screens. In Figure 3.8, we can see an example of a website screen. This screen refers to the homepage. We can see the login form and the buttons that take to the forget password and register screen. It can also be seen how the screen correlates to UED in terms of buttons, inputs and the navigation.

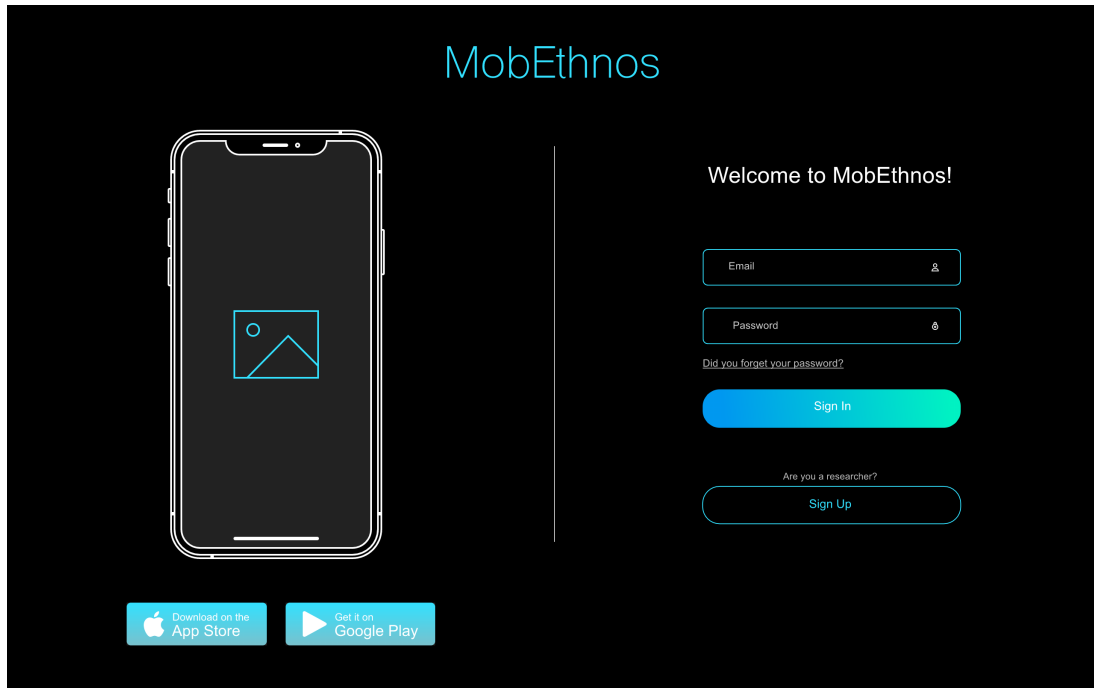


Figure 3.8: Homepage User Interface

In Figures 1, 2, 3, 4, 5, 6, 7, 8 and 9 from Appendix B - User Interface and in the Annex with the name Software Requirements Specification, the rest of the screens for the website can be seen.

In Figures 3.9 and 3.10, two examples of screens can be seen for the mobile application. They are the homepage and the project screen. The homepage screen is the first screen shown when an user enters the application for the first time. The project screen shows the touchpoints already submitted to that project, there is also a button for an user to leave the project and he can click on a button to add a new touchpoint which navigates to a different screen, the New Touchpoint Screen which is Figure 15 from Appendix B - User Interface. The rest of the screens made for the mobile application can be seen in Figures 10, 11, 12, 13, 14 and 15 from Appendix B - User Interface or in the Annex with the name Software Requirements Specification.

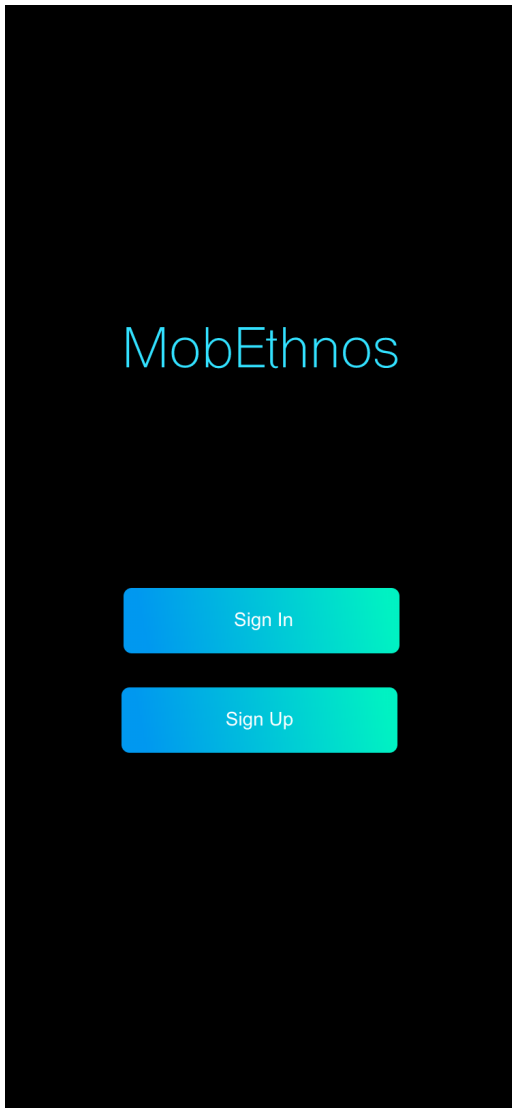


Figure 3.9: Homepage User Interface.

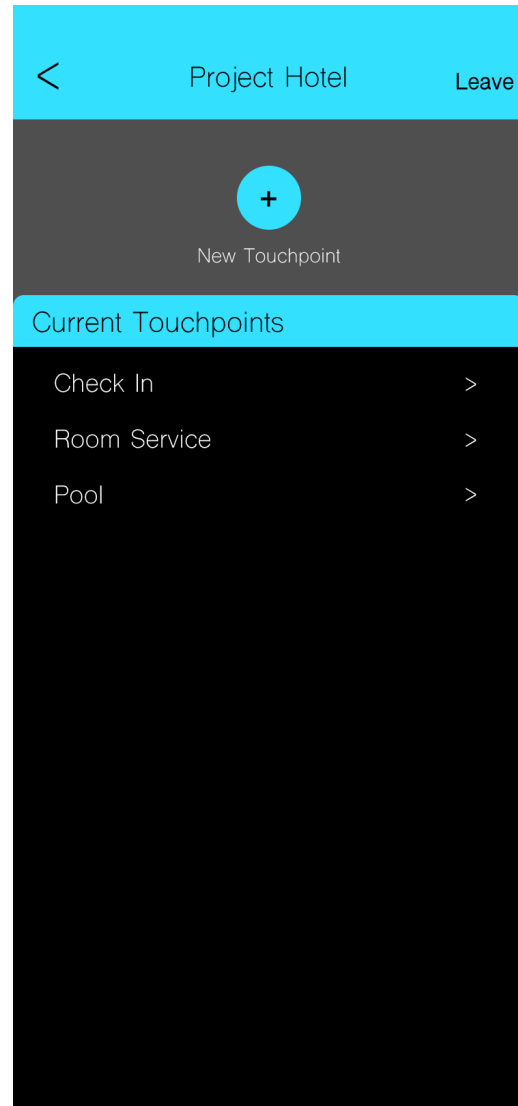


Figure 3.10: Project User Interface.

3.4 Non-Functional Requirements

The Non Functional Requirements of this project should provide the ability to make this software tool better for the end user. In order to reach these requirements, several discussions were taken with the advisor of this thesis and the most adequate ones were chosen.

The first and most important non-functional requirement should be usability. It consists of having a simple to use UI. The UI was made with this in mind. With usability testing, we will see if this requirement is accomplished.

Another non-functional requirement is scalability. With this requirement, we do not want to have a decrease in performance by having a class of 200 students using the application at the same time. Firebase already provides scalability, their database is indexed in a way to not have drops in performance.

Availability is also important. Deferred Syncing is already automatically implemented in the Firebase software development kit and provides availability, allowing participants to use the application even when an Internet connection is not available.

3.5 Functional Requirements Prioritisation

This section presents a list of requirements for the participant and the researcher. It contains the name of the requirement, the use case ID, and the MoSCoW prioritisation method for each one.

MoSCoW prioritisation divides the requirements into four specific priorities:

- **Must Have** - Mandatory requirements needed for the project to work.
- **Should Have** - Not mandatory, but adds relevant value to the project.
- **Could Have** - Nice to have, but if not implemented will have a small impact on the project.
- **Will Not Have** - Not to be implemented at this specific time, but can be implemented in the future.

3.5.1 Participant Requirements

1. **Sign Up** - ID: 01 - Must Have
2. **Sign In** - ID: 03 - Must Have
3. **Join a New Project** - ID: 05 - Must Have
4. **View Projects that you are Involved in** - ID: 06 - Must Have
5. **Change Password** - ID: 07 - Must Have
6. **Delete Account** - ID: 08 - Must Have
7. **View all Touchpoints Submitted** - ID: 09 - Must Have
8. **Edit Touchpoint** - ID: 10 - Must Have
9. **Add New Touchpoint** - ID: 11 - Must Have
10. **Leave Project** - ID: 12 - Must Have
11. **Logout** - ID: 13 - Must Have

3.5.2 Researcher Requirements

1. **Sign Up** - ID: 02 - Must Have
2. **Sign In** - ID: 04 - Must Have
3. **Change Password** - ID: 14 - Must Have
4. **Delete Account** - ID: 15 - Must Have
5. **Create Project** - ID: 16 - Must Have
6. **View Own Projects** - ID: 17 - Must Have
7. **Finalize Projects** - ID: 18 - Should Have

8. **Archive Projects** - ID: 19 - Should Have
9. **View Project Journeys** - ID: 20 - Must Have
10. **Export Data** - ID: 21 - Could Have
11. **Delete Project** - ID: 23 - Must Have
12. **Manipulate Project Journeys** - ID: 22 - Must Have
13. **Show Project Map** - ID: 24 - Should Have
14. **Show Overview Graph** - ID: 25 - Could Have
15. **Logout** - ID: 26 - Must Have

Chapter 4

Software Architecture

This chapter describes the software architecture. We use the C4 model to visualise it. This model "(...) is an ‘abstraction-first’ approach to diagramming software architecture, based upon abstractions that reflect how software architects and developers think about and build software" [9]. The hierarchy of the C4 Model is divided in four levels, Context, Containers, Components and Code. In our project we will only use Context, Containers and Components levels. The level Code will not be used due to being a rapidly changing environment and also to have more freedom to create the components when the platform is being developed. Another reason to not deepen our architecture even further is the fact that the complexity is not extremely high for our Components and if we need to generate the Code Diagrams for the Components, a lot of tools can do it after the system is complete.

A good sentence to understand the C4 Model is the following: "A **software system** is made up of one or more **containers** (web applications, mobile apps, desktop applications, databases, file systems, etc), each of which contains one or more **components**, which in turn are implemented by one or more **code** elements (e.g. classes, interfaces, objects, functions, etc)." [9]. Here, we can clearly see the division of the four levels of this model.

Starting with the first level of the C4 Model, Context Diagram, we can see the system as a whole which provides readability for all technical and non-technical people, inside and outside the organization. Regarding the architecture of our system, we included the actors, User and Service Provider or Researcher, the Software Systems, both internal and external and the relationships between them. We can visualise Google’s Firebase as an external software system and Costumer Experience Evaluation as an internal software system. Google’s Firebase is a Backend-as-a-service and provides ready-made APIs for helping in developing our software faster. Figure 4.1. shows the visualisation of the Context Diagram of the architecture:

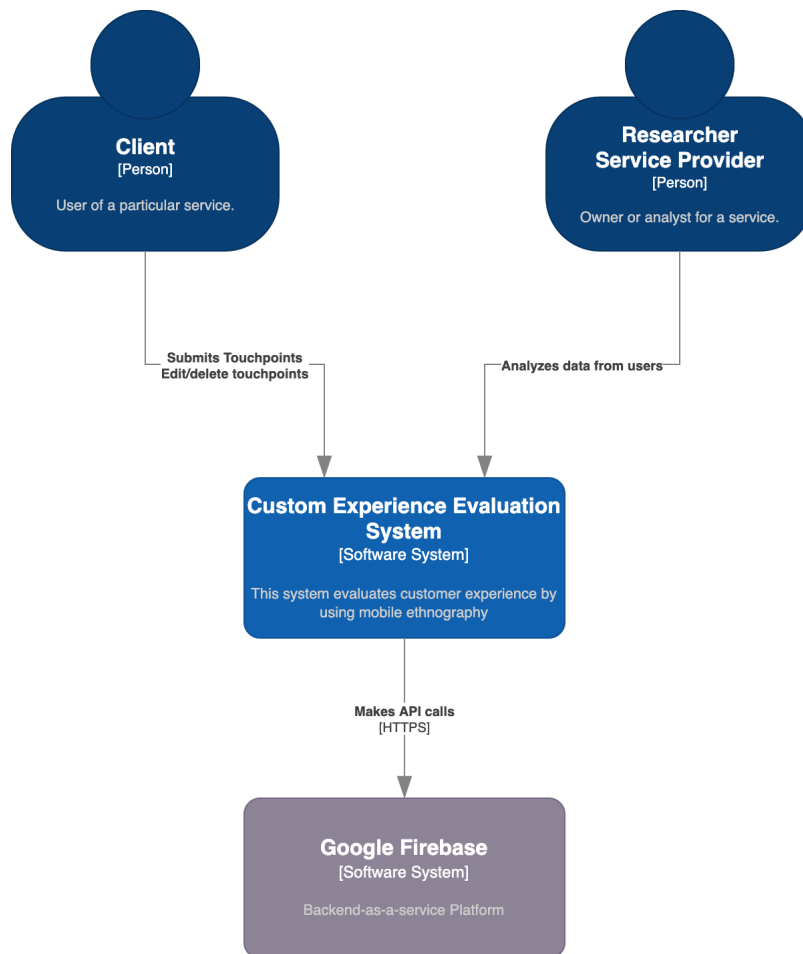


Figure 4.1: Context Diagram

The second level of the C4 Model is the Container Diagram. This one displays more detail about the software systems of the Context Diagram and also shows some more distributions across the system. However, it is still very high-level. The intended audience for this level is "Technical people inside and outside of the software development team; including software architects, developers and operations/support staff." [9].

Regarding our software tool, we can see the two main components that will be made, the progressive web application and the single-page application. We can also see another container called Web Application, which will serve the latter. Also, the mobile application container will serve the progressive web application. The technologies for these containers are also shown in this diagram. For the progressive web application, we will use React Native and JavaScript, for the single-page application, we will use React.js and JavaScript, and for the Servers (Web Application and Mobile Application) we will use Nginx.

The bottom layer of our Container Diagram shows how Google's Firebase works. We have several micro-services: Firebase Authentication, Cloud Firestore and Cloud Storage. They communicate with our applications via HTTPS and provide APIs with functions already done to deal with the several functionalities needed in our components. Each one of these micro-services has a storage or a database in the cloud. Figure 4.2. shows the Container Diagram for our architecture:

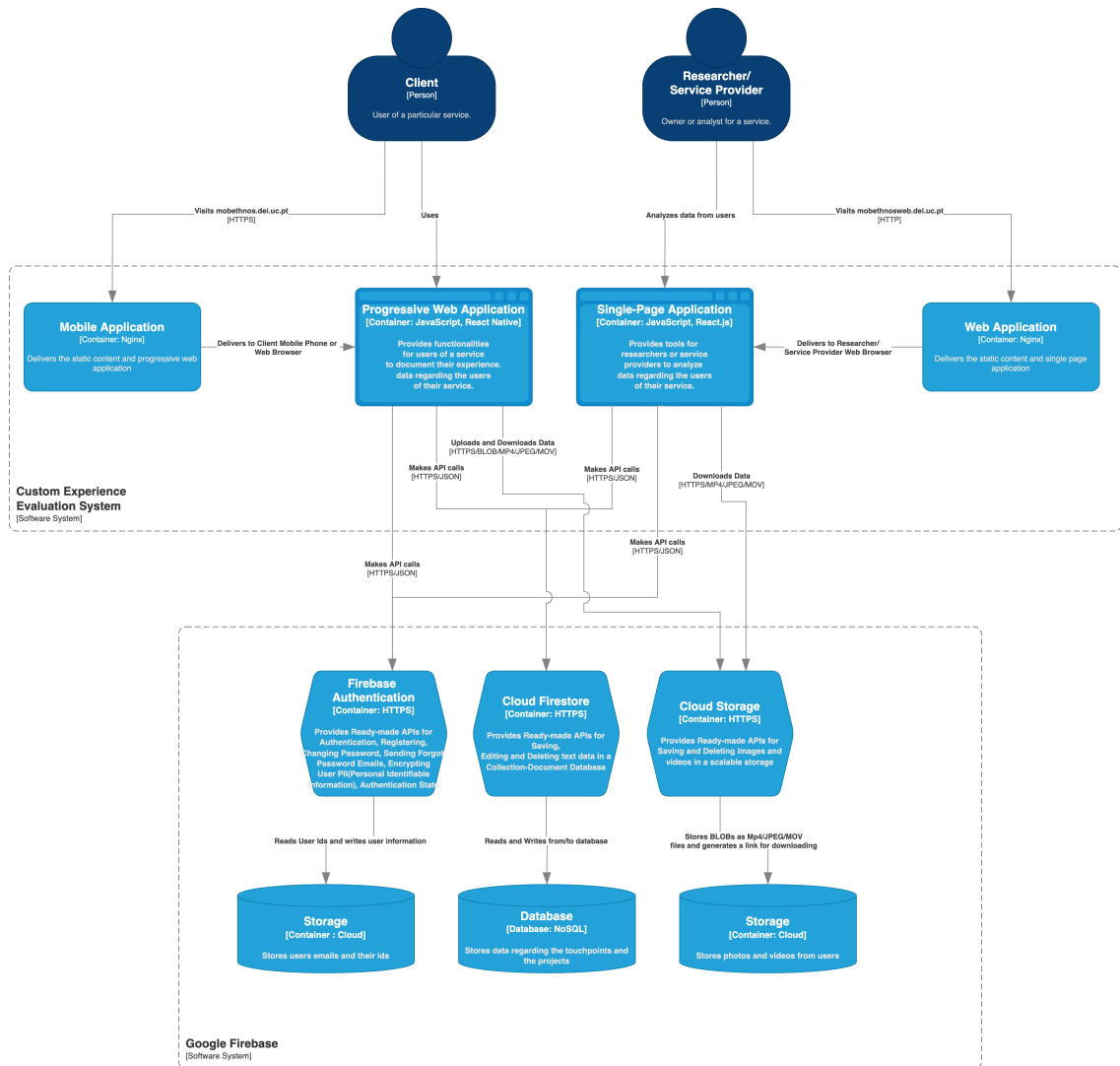


Figure 4.2: Container Diagram

The third level of the C4 Model is the Component Diagram. It shows the details of the Containers and is intended for the developers and for the software architects of the system. This level is not mandatory in the C4 model. However, having it creates value in our architecture by having an organisation for the various components to develop and to have a better documentation. It will also help to apply the scrum methodology by facilitating the division of work into several tasks prior to development.

Our architecture is organised into two main containers. The single-page application and the mobile application. These Containers are the only ones to be decomposed into Components as they are the ones to be developed using the technologies chosen in the state-of-the-art.

First, we have the Progressive Web Application Component Diagram. It shows a Controller for the authentication, which uses Firebase Authentication and it is done with a React Hook. We also have the several Components of the application, each one represents a screen of our Mobile Application. The technology for each one of these Components is React Native and JavaScript. These screens have been defined in the Software Requirements Specification document in Annex 2 and we have an abridged version in the previous chapter, Chapter 3 - Requirements. After this explanation of the Components, we can now describe the relationships. For these ones, we can see how each Component is using Google's Firebase services and how they connect using HTTPS via their ready-made APIs. There are several services of Google's Firebase and it is explicit in the architecture visualisation the functionalities of each Component in relation to the specific service in Firebase. One Component can use more than one service from Google's Firebase. We can also visualise that the only actor displayed in this diagram is the user, as he is the one to use the Mobile Application.

Second, we have the Single-Page Application Component Diagram. The organisation of this is similar to that of the mobile application. We decompose it into several screens of our application. There is a difference in this one, we do not have an authentication controller. The control for the authentication is done inside each Component, thus allowing to only go to the pages where authentication is needed if the researcher or the service provider is authenticated. This is because of the different libraries for navigation in React.js and React Native. In React.js we use React Router which allows us for creating different routes each one corresponding to a different component. In React Native we use React Navigation which allows us to have stacks of several components where the user can navigate and by having an authentication controller we can select what is the stack to load. We can also see the different actor, Researcher or Service Provider, regarding the single-page application container.

Figure 4.3. shows the progressive web application component diagram, and Figure 4.4. shows the single-page application component diagram. These ones will help us to pave the way into development.

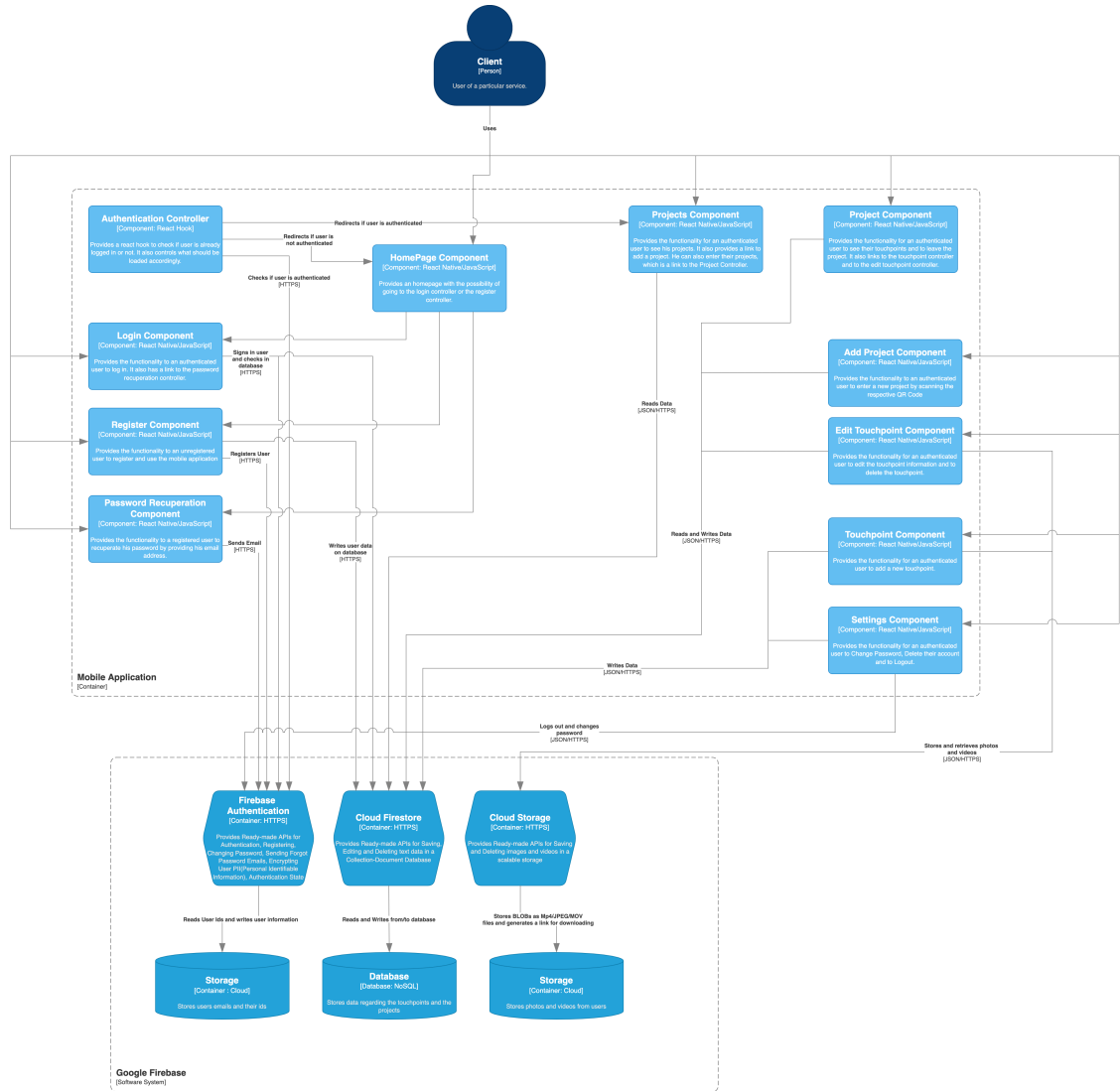


Figure 4.3: Progressive Web Application Component Diagram

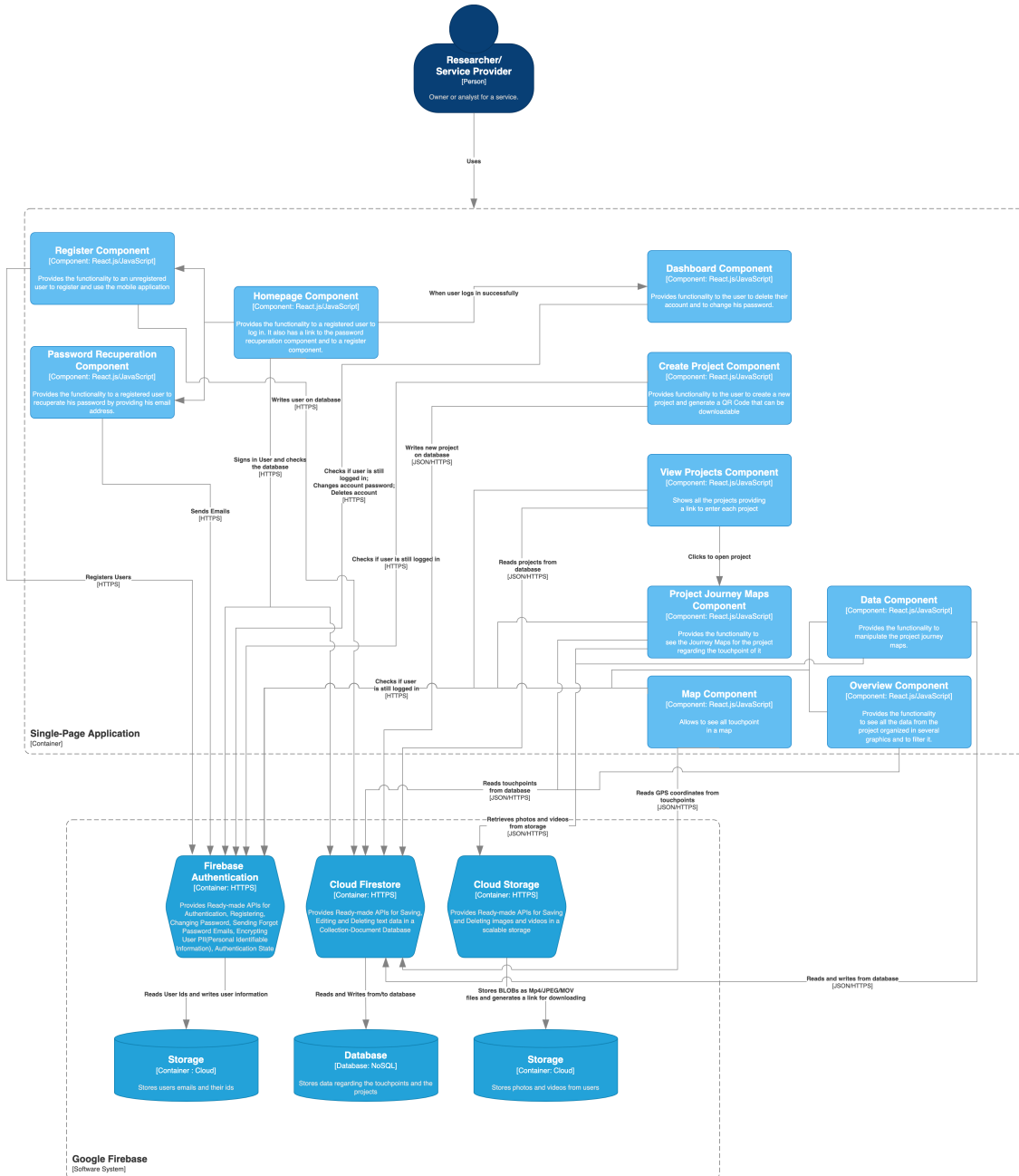


Figure 4.4: Single-Page Application Component Diagram

To conclude, the C4 model will help organise the development of our project. It will allow us to have a better explanation of our components and how they connect to the various services that our application uses. This will make our development faster, more concise and easier to understand by other technical people, i.e. software engineers, developers, software architects, etc. If someone wants to improve this project, he will be doing it with a quicker pace. One way to further complement our architecture is to use an Unified Modeling Language (UML) diagram. It is a better way to represent the use of external services. In the development chapter, we will explain in detail these external services, how they work with the front-end, and how they are structured.

This page is intentionally left blank.

Chapter 5

Software Development

In this chapter, everything about the development of our software tool will be detailed. The development process uses three environments:

- **Development** - This environment is on the computer of the author of this thesis. Any code updates will not affect the final product until this process is deployed into the production environment. GitHub will also be used for version control and to send code updates to staging. The website and the mobile application will run on the computer of the author of the thesis. For the mobile application, we will simulate a device using Chrome DevTools and will allow us to simulate a device in the browser. Preliminary testing will be done in development in an attempt to minimise bugs in the next environment.
- **Staging** - This environment is for more extensive testing, and all testers will use it. It is not on the local machine, but instead on a remote server, and the mobile application can be downloaded into the testers' devices. Regarding the website, GitHub will be used for automatic deploying it on the Firebase Hosting service. For this, a workflow was made on GitHub that allows for an automatic deployment whenever a push action is made. For the mobile application, we will deploy the PWA using Netlify. This service has free servers available and all that is needed is putting the files of the PWA, i.e., after the project has been built by Expo, on the server. With this, testers can install the app on their device for a close to production experience.
- **Production** - This environment is for when all testing is done and we are ready to launch our software tool to the public. The website and the PWA will be served on a server from DEI. The mobile application can be installed on the users' devices.

5.1 Back-end Structure

In this section, the Firebase structure for the various services will be explained. These include authentication, the Firestore database and Firebase storage.

5.1.1 Firebase Authentication

This service from Firebase is used to register, log-in, and recover the password of all users of our software tool. Using the libraries provided by Firebase, more specifically *firebase/auth*, the user uses the email address and password to register. After this, a user will be created with a unique identifier that will later be used to store data related to him in the database. In addition, the log-in functionality is also available by using the corresponding libraries. In conclusion, the password recovery functionality will also be ready to use after the user has registered. By calling the appropriate function of the libraries, the user will receive an email asking him to enter a new password. The body of the email can be edited in the Firebase console according to our needs.

5.1.2 Cloud Firestore

Cloud Firestore is a NoSql, document-oriented database. It is used to store all the text data and links of photos and videos. The structure for this is divided into three main collections:

1. ***Participants*** - It has a document for each participant and also controls the log-in functionality. Only allows participants to log in on the mobile application. Each document is made up of user id, birth date, gender, and name. In addition, it contains a collection of projects in which each user is involved.
2. ***Researchers*** - It has a document for each researcher and also controls the log-in functionality. Only allows researchers to log-in into the web application. Each document is composed of the user id.
3. ***Projects*** - It has a document for each project containing the name, the researcher id, and the terms for that particular project. Inside each document, it contains three collections:
 - (a) *tags* - It has a document for each tag added by the researchers to the project.
 - (b) *touchpoints* - It has a document for each touchpoint added by a participant. Inside each document it contains all the information regarding the touchpoint.
 - (c) *users* - It has a document with the id of the participant to know what users are inside that project.

In figure 5.1., a schema for the database can be seen, where the yellow boxes represent collections and the blue boxes represent documents inside the collection. Note that a document can have collections associated with it and a collection is comprised of documents.

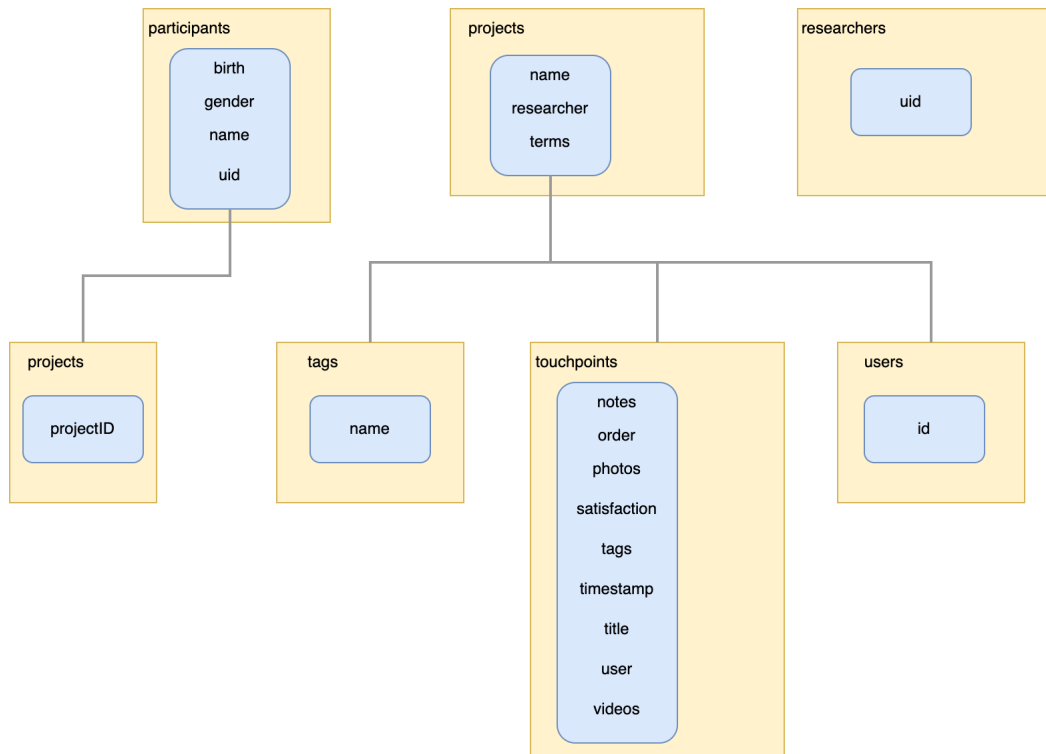


Figure 5.1: Cloud Firestore Database Schema

5.1.3 Firebase Storage

Firebase Storage is a cloud storage service that allows to store static content. In our case, it will store photographs and videos related to the touchpoints. Using the libraries provided by Firebase, we can upload the content and subsequently a link for it is provided. The link is then stored in the Cloud Firestore database.

The structure of the folders is very straightforward. We have a folder called projects which will store all the data relative to the projects. Inside this folder, we have a folder to each project and inside each project folder, we have a folder for each user. Thus, allowing a simpler process if we want to eliminate all data from an user or a project. In Figure 5.2., we can see the folder structure in more detail. Note that the folder that contains all the projects can have an unlimited number of them and a folder of a project can have an unlimited number of users.



Figure 5.2: Firebase Storage Folder Structure

5.1.4 Security

Regarding the security of these Firebase services, some rules were written to try to remove unwanted users. For Firebase authentication, there can only be new 100 users per hour using the same IP address. This protects our project from misuse and malicious users. We reached this number due to one IP address of a company can have many of users but generally not more than 100, it is just an estimate. Also, this number can always be changed. For the firestore database and firebase storage, a rule was written to only allow reads and writes from authenticated users. In this way, even if users have the API keys, they cannot delete and write data from the database unless they are authenticated.

5.2 Mobile Application Development

In this section, the development of the mobile application will be explained in every step of the process.

5.2.1 Creation of Files and Installation of Modules

The first step taken in the mobile application was to create all the files and install all the modules in order to start the development. First, Node.js and npm should already be installed on the computer. Then, expo SDK was installed with the following command in the terminal:

```
npm install --global expo-cli
```

After the installation, the project was created with the following command:

```
expo init mobethnos
```

As the project has been created, we will analyse it and explain the various parts of it. Going to the root of the project, we can see a file called *App.js*, this is the starting page for the application. A file called *app.json* can also be seen, which contains some API keys and project information such as names, icons, and splash images. The file structure will be divided in three main directories:

- *components* - Includes all the screens of our application.
- *config* - It has the configuration to connect our application to the back-end.
- *assets* - It stores the static content of the application, i.e., the images required for the screens.

Regarding other directories, we also have *node_modules*, this one contains all the modules installed via npm, in other words, the open source libraries for React Native.

5.2.2 Connection to Firebase

To connect our application to Firebase, the first step taken was to create a new project on Google's Firebase console. After this, a set of API keys was given to us. In the *config* folder of our project, we created a file called *firebase.js* that contains the configuration. Prior to that, the libraries for Firebase were installed. Also, for deferred syncing and for working offline seamlessly, we added one function of Firebase library, *enableIndexedDbPersistence()*, it persists and caches data from the Cloud Firestore database. The configuration file looks like this; however, the API keys shown here are substituted with the symbol #:

```
import { initializeApp } from 'firebase/app';
import { enableIndexedDbPersistence, getFirestore }
from 'firebase/firestore';

const firebaseConfig = {
  apiKey: "#####",
  authDomain: "mobethnos.firebaseio.com",
  projectId: "mobethnos",
  storageBucket: "mobethnos.appspot.com",
  messagingSenderId: "#####",
  appId: "#####:web#####",
  measurementId: "#####"
};

const app = initializeApp(firebaseConfig);

const fs = getFirestore();
const db = enableIndexedDbPersistence(fs);

export default app;
```

5.2.3 Authentication and Navigation

Regarding the navigation, a library called React Navigation [15] allowed us to simplify it. Looking at how it works, we can have multiple stacks of components to render given a condition. By using a listener from the Firebase Authentication library we can see the authentication status and render the matching stack. To listen for this a new React Hook was created, the following code shows how this hook works.

First, we have the hook itself to listen to the status of the user and then return it.

```
import React, {useState} from 'react';
import { getAuth, onAuthStateChanged } from 'firebase/auth';

const auth = getAuth();

export function useAuthentication() {
  const [user, setUser] = useState("loading");

  React.useEffect(() => {
    const unsubscribeFromAuthStatusChanged = onAuthStateChanged(
      auth, (user) => {
        if (user) {
          setUser(user);
        } else {
          setUser(undefined);
        }
      });
  });

  return unsubscribeFromAuthStatusChanged;
}, []);

return {
  user
};
}
```

Second, we have the component that will render the stack corresponding to the user status, the UserStack is the authenticated one and the AuthStack is the unauthenticated one. The hook made previously will return the user status to this component. In addition, this component will be called on *App.js*, which is the first component to be rendered in the application.

```
import React from 'react';
import { ActivityIndicator } from 'react-native';
import { useAuthentication } from './useAuthentication';
import UserStack from './UserStack';
import AuthStack from './AuthStack';

export default function RootNavigation() {
  const { user } = useAuthentication();

  if (user === "loading"){
```

```

    return <ActivityIndicator style={{position: 'absolute',
    justifyContent: 'center', alignSelf: 'center', left: 0,
    right: 0, top: 0, bottom: 0}}/>
  }
  else {
    return user ? <UserStack /> : <AuthStack/>
  }
}

```

Finally, we have the components containing all the screens of the navigation, the AuthStack and the UserStack mentioned above. In the UserStack, we have all the the components with the functionalities for an authenticated user. These include viewing all projects, adding a new project, seeing, adding, and editing touchpoints, and a settings component that contains the functionality to change password, delete account, and logout. We can see the code for the UserStack in the following.

```

import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

import Projects from './Projects';
import AddProject from './AddProject';
import SettingsScreen from './SettingsScreen';
import ProjectScreen from './ProjectScreen';
import TouchpointScreen from './TouchpointScreen';
import EditTouchpoints from './EditTouchpoints';

const Stack = createNativeStackNavigator();

export default function UserStack() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Projects"
          component={Projects} .../>
        <Stack.Screen name="AddProject"
          component={AddProject} .../>
        <Stack.Screen name="Settings"
          component={SettingsScreen} .../>
        <Stack.Screen name="Project"
          component={ProjectScreen} .../>
        <Stack.Screen name="Touchpoint"
          component={TouchpointScreen} .../>
        <Stack.Screen name="Edit"
          component={EditTouchpoints} .../>
      </Stack.Navigator>
    </NavigationContainer>
  );
}

```

In the AuthStack, we have the functionalities regarding the unauthenticated user. These

include the home screen, the login, the registration, and the password recovery. We can see the code for the AuthStack in the following.

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
 '@react-navigation/native-stack';

import Login from '../components/Login';
import Register from '../components/Register';
import HomeScreen from '../components/HomeScreen';
import PasswordRecuperation from './PasswordRecuperation';

const Stack = createNativeStackNavigator();

export default function AuthStack() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName='Home'>
        <Stack.Screen name="Home"
          component={HomeScreen} ... />
        <Stack.Screen name="Login"
          component={Login} ... />
        <Stack.Screen name="Register"
          component={Register} ... />
        <Stack.Screen name="PasswordRecuperation"
          component={PasswordRecuperation} ... />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

5.2.4 Unauthenticated User Functionalities

The unauthenticated user functionalities are the log-in, the registration, and the password recovery. In this subsection, we are going to explain how these functionalities work.

The registration screen consists of a set of inputs. These inputs include name, birth date, sex, email address, and password. Figure 5.3. shows the screen for this functionality. The other unauthenticated functionalities follow the same logic as the register screen. They have a set of inputs and subsequently they are sent to the back-end by Firebase's functions.

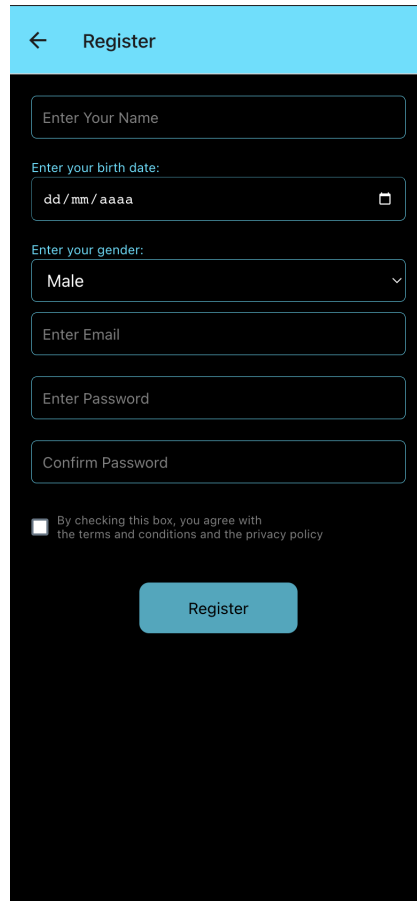


Figure 5.3: Register Screen of the Mobile Application

When a user fills in the input on this screen and clicks the register button, it will sign up the user with the appropriate function and subsequently write this data in the database. It also marks this user as a participant. Thus, not allowing one to log-in as a researcher. If a user is already registered, the system warns the user and doesn't let him register.

Regarding the log-in functionality, the user enters the email address and password. When clicking the log-in button, the system logs the user in using the appropriate function from the libraries. If the account associated with the user that is trying to log-in is a researcher one, the system sends a warning and logs the user out.

The password recovery functionality comprises an input for the user's email address. When clicking the button to recover the password, the system sends an email to the user to change his password. This works by calling the Firebase library function, *sendPasswordResetEmail()*, and defining the email to send to the user in the Firebase console. After the user has changed his password, he can log-in with the new one immediately afterwards.

5.2.5 Authenticated User Functionalities

The authenticated user functionalities are the essence of our application. These include seeing all projects, adding and leaving a project, adding, editing, and eliminating touch-points, and also changing password and deleting the account.

Functionalities for the Screen that Contains all Projects

When a user logs in, the first screen that he sees has the projects that he is a participant in. The functionalities are: seeing all the projects, a button to go to the screen to add a new one, and a button to go to the settings screen. It also has a functionality to submit touchpoints to the database that were uploaded when the user was offline. In Figure 5.4., we can see the screen that contains these functionalities. In this particular case, it contains two projects: *Projeto Teste* and *Projeto Ativo*. We can also see that there are no touchpoints that were submitted offline.

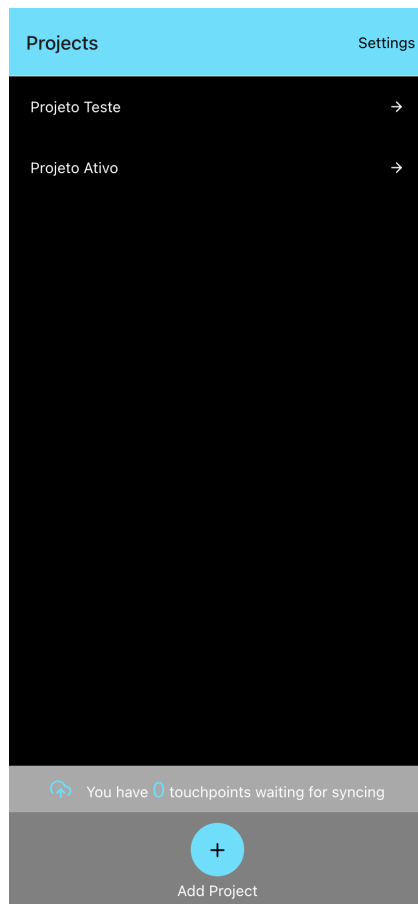


Figure 5.4: Projects Screen of the Mobile Application

To get all the projects in which a user participates, we call a function in the `useEffect` to go to the back-end to fetch these data. The `useEffect` is a hook for React, this hook tells the component to do something when the screen renders. It can be used to fetch data or update the DOM. Every time we need to get something when the screen renders we use this hook.

Regarding the function to get the projects, the collection of the projects for the logged in user is read from the database and afterwards the data for each project is read. If the status of the project is active, we add the data of the project to an array. The data include the name and id of the project. We then set that array as the state for the projects so that we can pass it to the flatlist containing all the projects. For setting the state, we created a new state called `projects` in the beginning of the component with the `useState` hook. This allows us to have a function with an initial value that can be changed and later used in the render of the component. This hook works just like a global variable. Also, when the screen is rendered, the projects are always updated. If the user clicks on a project,

it goes to the screen where he has all the information submitted to the project that will be explained later. It does so by using the React Navigation library. The name and id of the project and the user id are also passed to the screen of each project. We can see an example of how it works in the following code snippet:

```
<TouchableOpacity ... onPress={() => syncing ?
    setShowAlert(true):
    navigation.navigate("Project",
    { name: item.name, id: item.id, user_id: uid })
  }>
...
</TouchableOpacity>
```

The `TouchableOpacity` corresponds to a button, which in this case shows the project. The syncing state is related to the functionality of deferred syncing which will be explained ahead.

In order to navigate to the screens to add a new project and settings, the React Navigation library is used. The `navigate` function is called, which derives from the navigation called from the props. These props are defined in the Stack for the navigation. Props are used to pass data between components and from parent to child components.

Lastly, the function to submit the touchpoints that a user has submitted while offline works by using Localbase. It works like firestore database with a key and value but uses IndexedDB from the browser. For a more detailed explanation of Localbase, we can see the Github page [11] of the creator regarding this library. Another library, NetInfo [10], allows to see if the user is connected to the internet. If this is true, the touchpoints in this offline database are written in the firestore database and the static content, photos and videos, uploaded to firebase storage. When uploading to the back-end is being done, the user cannot leave this screen and, if he tries, is warned about losing information about the touchpoints. In the previous code snippet, a state called `syncing` could be seen. It indicates if the content is being uploaded from the offline to the online database.

Functionalities For the Screen For Each Project

The screen for each project contains the functionalities to view all the touchpoints submitted, leave the project, and navigate to the screen to add a new touchpoint. In Figure 5.5., this screen can be seen. In this Figure, three touchpoints are provided as an example: *Touchpoint 1*, *Touchpoint 2* and *Touchpoint 3*. These are in descending order in terms of time. This means that we can first see the most recent and then the latter.

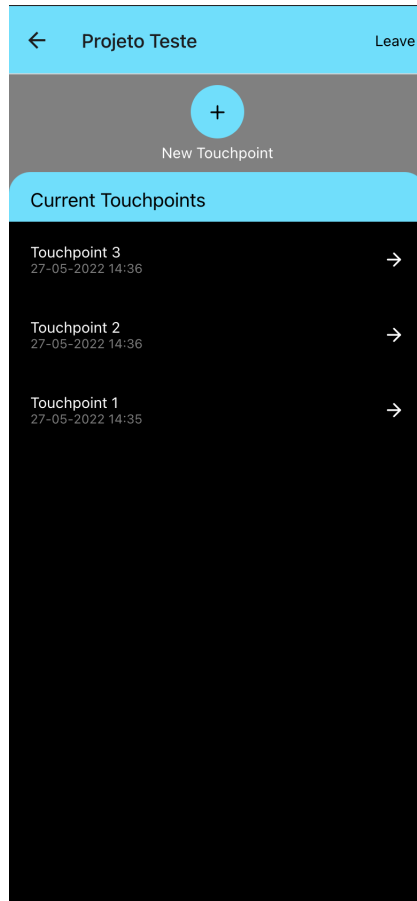


Figure 5.5: Project Screen of the Mobile Application

Starting with navigation functionalities, such as going to the screen to add a new touchpoint and going to each touchpoint screen, the React Navigation library is used in a similar manner to what was explained above. For going to the screen of each touchpoint, the data regarding to the touchpoint is passed on to it by using React Navigation.

Then, we have the functionality to see all the touchpoints submitted. Similarly to the one on the Projects screen where we could see all projects, this one is also inside a `useEffect` hook. Inside this hook, there is a function to fetch the touchpoints of a particular user in a particular project. After getting them, we sort them by descending order, set the state, and pass it on to the corresponding Flatlist.

Finally, to leave the project, the user clicks the button and a function for this is called. Inside this function, all photos and videos of this user in this project are deleted by going to that specific folder in the firebase storage. After this is done, all the documents in the firestore database that connect this user to this project are also deleted. The system then redirects the user to the screen that contains all projects.

Adding a Touchpoint

In this functionality, we can add a new touchpoint to a specific project. Each one contains a title, satisfaction, notes, and then optional photos, videos, and location. In Figure 5.6, the screen can be seen regarding this functionality.

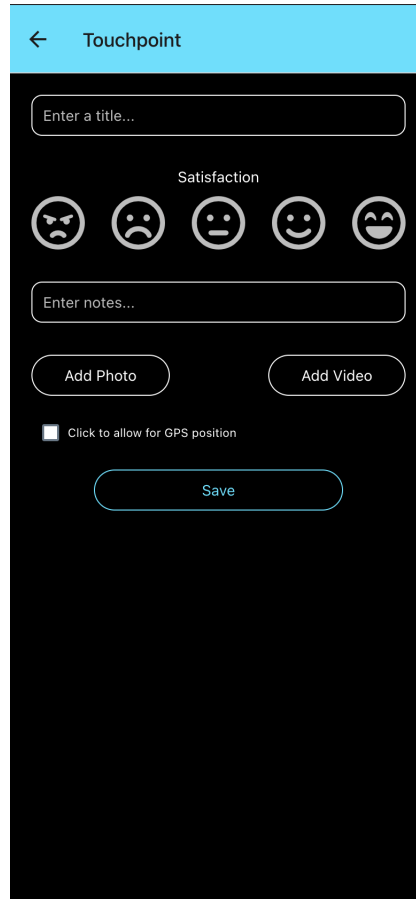


Figure 5.6: Add Touchpoint Screen of the Mobile Application

When the user clicks on the save button, the system checks if network connection is available by using NetInfo library. The system also shows a small notification in the bottom of the screen warning the user if he has internet connection or not. This can be seen in Figures 5.7 and 5.8.

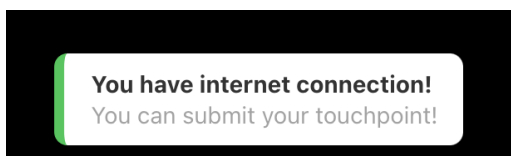


Figure 5.7: Successful Internet Connection Notification

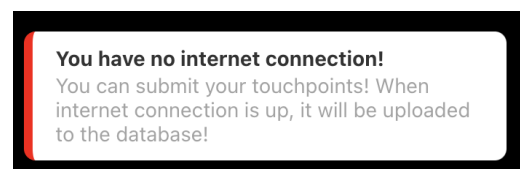


Figure 5.8: Unsuccessful Internet Connection Notification

In this functionality, there are two distinct parts. If the user has internet connection, the data is saved in firebase storage and firestore database. If not, it is saved using the Localbase offline database.

Starting with the on-line part, when the user clicks the save button, photos and videos are stored in firebase storage. Subsequently, a link for each one is retrieved. Afterwards, the

data for each touchpoint and the links are written in the database.

Moving now to the offline part, we create a new collection in Localbase called *touchpoints*, where we add an object containing all the data submitted. Photos and videos are stored in base64 format. When this is done, the data is saved offline and when the user enters the application again with network connection the touchpoint is submitted as stated above in **Functionalities for the Screen that Contains all Projects**.

Some libraries from Expo were used, expo-image-picker to get photos and videos and expo-location to get the location. Another library called moment was also used to get the current timestamp.

Edit Touchpoint

The edit touchpoint functionality has the same information as adding a new one; however, the user cannot edit the location where it was submitted, or if he did not add the location, he cannot add it posteriorly.

When the user opens the touchpoint screen, he can click on notes, satisfaction, and title to edit it. By doing so, a modal will be opened, and then he can insert the new information and save it. All the information will be saved in a useState hook, which will allow to immediately update after the change is made. This speeds up the front-end by showing instant updates. After the user has saved the information inserted in the modal, the corresponding state is updated and the system uses firestore libraries to update the information in the database.

Regarding photos and videos, if the user deletes one of them, this content will be deleted from the firebase storage and then from the database. After this is done, the state will be updated by removing the content. For adding static content, the logic is the same as when adding a new photo or video while adding a new touchpoint. First, the content is stored in firebase storage and a link is retrieved. Then, the link is stored in the firestore database.

Finally, the user can delete a touchpoint. When this happens, the static content is firstly deleted from firebase storage using the corresponding links and, afterwards, the data of this touchpoint is deleted from the database. Finally, the system redirects the user to the project screen using the React Navigation library.

Functionalities for the Settings Screen

In the settings screen, the user can change the password, delete the account and log-out.

To change the password, the user has to enter his password, the new password, and confirm this new password. When the user clicks the change button, the system re-authenticates the user and retrieves a credential using the firebase auth libraries. This function is asynchronous and has a try catch inside. If something goes wrong, there is an alert inside the catch that will warn the user. So if the firebase auth library can not authenticate the user, an alert will be made and the function will warn the user. If all goes correct and the user chooses a strong password according to firebase auth parameters, the password is changed. If the password is not strong enough, the system will warn the user.

To delete the account, the user has to enter his password. It works with a credential, re-authentication, and a try catch, like the function to change password. However, in this case, all user data will be deleted. To do so, we first loop through all projects of the user to

delete the user data related to the project. Inside this loop, we are going to each firebase storage folder corresponding to the project that is being looped with the user id. The path to the storage looks like this:

```
ref(storage, 'projects/${document.data().projectId}/${uid}');
```

Inside this folder we are going to delete all the content. Then, we are going to delete all the touchpoints corresponding to the user, by going to the appropriate location in the firestore database. In the final part inside the loop, we go to the projects collection inside the participant document and delete it and also, delete the user from the users collection inside the project document. This loop is done for all projects. After all projects have been deleted, we delete the document corresponding to the user in the participants collection and finally delete the user using the firebase auth libraries. As there is no user signed-in, the React Navigation library will render the stack corresponding to the unauthenticated functionalities.

5.2.6 Progressive Web App Requirements

For a PWA to work as intended and to be installed on the users' devices, there are three requirements that must be met [35].

The first one is that the website must have a web manifest file. This requirement is automatically met using Expo SDK. When the website is built, it automatically creates this for the website. The manifest allows the system to know several details, such as splash screen, the icon and name for the PWA.

The second one is that the website must run be served using HTTPS. It "provides a layer of security and trust between the application and the browser" [35].

The third one is that the website must register a service worker. "The service worker provides the extensible backbone for event-driven functionality to execute on a separate thread from the user interface." [35]. To create the service worker, @expo/webpack-config and workbox were used. First, we need to run expo customize:web to create the *webpack.config.js* file. Then, using the expo tutorials, we created the file *service-worker.js*, which is the service worker itself. The code to create it is provided by Expo in the following link <https://github.com/expo/examples/blob/master/with-workbox/src/service-worker.js>. To create the functions to posteriorly register the service worker, the file *serviceWorkerRegistration.js* was created and the code is also provided by expo in the following link <https://github.com/expo/examples/blob/master/with-workbox/src/serviceWorkerRegistration.js>. Finally, in the body of the *App.js* file, we register the service worker by calling the appropriate function from *serviceWorkerRegistration.js* which in our case is the *register()* function.

5.3 Website Development

In this section, the development of the website will be explained in every step of the process.

5.3.1 Creation of Files and Installation of Modules

To create the website, the process is similar to that of creating the mobile application. The requirements are the same; however, instead of using npm, we are going to use yarn, which is another package manager. The other requirement is that Node.js needs to be installed on the computer.

To create the project, the following command was typed in the terminal:

```
npx create-react-app mobethnos
```

With this we can start developing our project as we now have all the files and modules for a blank React.js website.

5.3.2 Connection to Firebase

To connect the website to Firebase, the process is similar to the one in the Mobile Application due to the frameworks being very similar, practically the same. The difference from the Mobile Application is that the *firebase.js* file is placed inside the components folder instead of creating a separate folder. Another difference was that we did not include the *enableIndexedDbPersistence()* function as a consequence of the requirements that state that the website does not need to work offline.

5.3.3 Navigation Between Components

For the navigation between components, a library called *react-router-dom* was used. It allows to assign a path to a component. Therefore, to navigate between pages, which are some of the components of the website, functions from this library were used.

To assign the components to paths, they were imported to the *App.js* file and assigned to a Route, which we can see in the following code snippet as an example:

```
<Router>
  <Routes path="/" element={<App />}>
    <Route index element={<WelcomeScreen />} />
    <Route path="forgetpassword"
           element={<ForgetPasswordScreen />} />
    <Route path="register" element={<RegisterScreen />} />
    <Route path="home" element={<Homepage />} />
    <Route path="createproject"
           element={<CreateProjectScreen />} />
    <Route path="viewprojects"
           element={<ViewProjectScreen />} />
    <Route path="project" element={<ProjectScreen />} />
  </Routes>
</Router>
```


In this example, the first screen corresponds to the component called WelcomeScreen which has the path "/" due to having the index prop and the path "/" being defined in the Routes component. Then, we can see more screens related to other components. For instance, we have the CreateProject component that has the path "/createproject".

5.3.4 Authentication

As the navigation with react-router-dom differs from React Navigation, the way components are loaded is different. When we enter on the website, the WelcomeScreen component is always rendered. Then the user can log-in; if he does so, the react-router-dom library redirects to the Homepage component. However, if the user tries to enter the components that need authentication and he isn't authenticated, the system will automatically redirect to the WelcomeScreen component. This works with a useEffect inside each authenticated component. First, a state for the user with an empty object is defined, then, in the useEffect a function to get the user status is called:

```

async function getAuthStatus() {
  const unsubscribe = await onAuthStateChanged(auth,
    (currentuser) => {
      setUser(currentuser);
    });

  return () => {
    unsubscribe();
  };
}

```

This is a function that will define the status of the user. Lastly, in the render part, if the user state is null or undefined, the Navigate function from react-router-dom is called to render the WelcomeScreen. Else, it renders the corresponding screen. An example can be seen with the following code:

```

if (!user) {
  return <Navigate to="/" />;
} else {
  return ( ... )
}

```

5.3.5 Unauthenticated User Functionalities

The unauthenticated user functionalities are similar to those of the mobile application, except for the addition of a button that redirects the user to the PWA link. These functionalities are log-in, registration, and password recovery.

In the log-in, a function is called that signs the user with the email and password. We used a function from Firebase auth library called *signInWithEmailAndPassword()*. After the user logs-in with success, the system verifies in the database if the user is indeed a researcher. If not, the system warns the user that he is a participant and does not let him log-in. This was a decision taken after the requirements. This is because in the future, if we need to commercialise our tool and researchers need to pay a subscription, a id from the authentication can be associated with a stripe id, for example. Stripe is a service that

allows payments and can be implemented in a commercialized version of our product [46]. If he is indeed a researcher, the system redirects to the homepage of the authenticated functionalities.

Regarding the registration, the user only needs to enter his email, password and confirm it. Unlike the mobile application, the system does not need to save additional information about the researcher as the name, age or gender of the researchers is not relevant to the analysis of a service. After the registration is successful, a document with the id of the researcher is inserted in the researchers collection in the Firestore database. This will prevent the researcher to log-in with the same account in the mobile application. This was a decision taken, as mentioned above, thinking about the future commercialisation of the tool and so, separate the accounts from researchers and participants by different users' emails.

To recover the password, the researcher only needs to enter his email, and subsequently an email will be sent to his address to redefine his password. This works the same way as for the mobile application. A function *sendPasswordResetEmail()* will be called with the user's email, and the back-end treats this automatically.

5.3.6 Authenticated User Functionalities

The authenticated user functionalities of this website are divided into several components. When the researcher logs-in, he has a homepage where he can change his password and delete his account. Other functionalities in the same dashboard include the creation of a project and seeing all the projects. The projects are divided by their status. When the user clicks on the project, he can see the journey maps, the maps, an overview, can add tags, delete the project, and customise the touchpoints to their needs by adding tags as stated in the requirements.

Dashboard functionalities

Starting with the password change and the deletion of the account functionality, in terms of code, is equal to that of the mobile application. The system re-authenticates the user with the current password and a credential is retrieved to subsequently call the function to change password or to delete the account. To change the password, we used the function *updatePassword()* from the Firebase auth library. To delete the account, we first loop through the several projects of the user. Inside this, we are going to go through all users of the project one by one. Then, we delete all static content from storage and delete the project document inside the participants collection in the document of each user. After going through all project users, we are still inside the loop of the projects. Here, we are going to delete the collections of touchpoints, tags, and users, and finally, delete the document of the project inside the projects collection. Then, moving out of all loops, we delete the document corresponding to the researcher account in the researchers collection and finally, delete the user using the *firebase/auth* library.

Create a project

For creating a project, a user has to insert the project name and the terms of the project, which can be a project description and what will the data be used to. For the terms of the project, we used a library called Mantine [31]. This library provides components with

very good usability and ease of implementation. In the Mantine library, we chose the RichTextEditor, which allows to create rich text by just using the component and pass the value of it to a state to later store in the database. To add to the database, the function from the firestore library is used and we define the status of the project as active. After the project is created, a QR code can be downloaded, and we can also see the token of the project on the screen. This token will serve as a way for participants to enter the project. This can be by inserting it on the proper input or by scanning the QR code. The QR code was created using the react-qr-code library, the value of which being the token of the project.

Viewing and Selecting Projects

Regarding the projects, they are divided into active, finalized and archived. To do this division, we used another component from Mantine, called Accordion. This provides windows for each project and can be opened when the user clicks on them. To open a project, the user just needs to click on top of them in the screen and will automatically redirect the user to the page of the project. The library react-router-dom was used to pass the project id to the component of the project itself. We did that by surrounding each element of the project with the Link component from the library. Each link leads to the path `"/project"`, and the project id is passed on to it by using the prop called state.

Functionalities in each Project Page

Inside each project page, we have a dashboard that is controlled by tabs. For this, we used a component from Mantine called Tabs. Each tab has a corresponding component. To find which project is to be loaded, the `useLocation()` function from the react-router-dom is used. Inside the location, we can check the value of state and retrieve it.

First, we have the Journey Map. This component contains a mix of libraries. For drag-and-drop, react-beautiful-dnd was used. For the elements of the screen, Mantine was used. In addition, Material UI and other libraries like Google Maps, React Player were also used. When the page renders, we get all the touchpoints and users of a specific project. When we load the touchpoints, we do it in a way that each user has their touchpoints. In figure 5.9., we can see the structure of the created array:

Our array will contain an object for each user. This object will contain the name, the age, the gender, the user id, and an array for the touchpoints of each user. It should be noted that this array of touchpoints is sorted by the order defined in the database coming from the drag-and-drop functionality or by the timestamp. Also, if the array containing all the touchpoints of the user is empty, the object will not be added to the array containing all the objects of each user.

When using the react-beautiful-dnd, we surrounded the container that includes all the drag-and-drop zones with a DragDropContext component. When a drag event ends, a function, `dragEndHandler()`, is called. Each container that includes the array of the touchpoints of each user is a Droppable component. This Droppable component as has a droppableId, which is the id of the user. Each element inside this Droppable component is a touchpoint which is surrounded by a Draggable component where we pass a key and a draggableId. These correspond to the id of the touchpoint. Also, an index is passed which is the position of the touchpoint in the array. In the function `dragEndHandler()`, we have a source and a destination. If the destination is null or if the droppableId of the source and destination

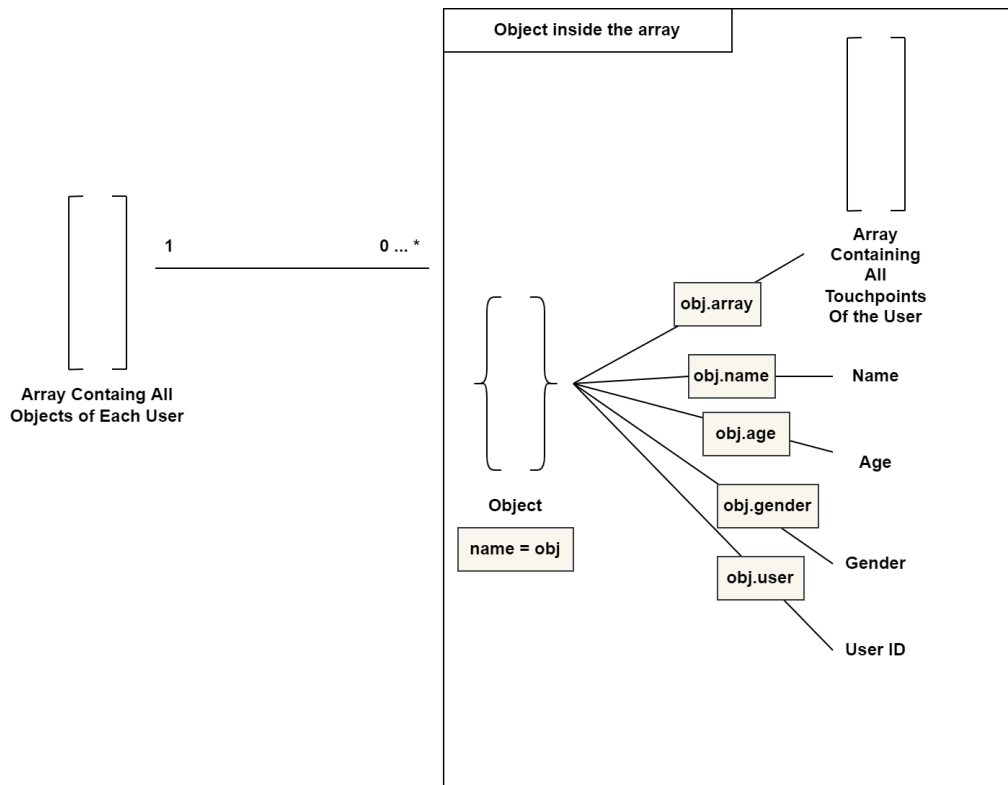


Figure 5.9: Structuring of Data for the Touchpoints Array

is different, then nothing happens, and the touchpoint will return to the initial position. If this does not happen, we are going to the array that contains all the objects of all the users, find the user and then change the order of the array accordingly. After that, we take the array of touchpoints of the user that we changed and create a new variable called order in the database in the document of each touchpoint. The order variable in the document of each touchpoint corresponds to the index of that element in the array.

Furthermore, the user has the option to add blank spaces between the touchpoints. These blank spaces are treated in the same way as touchpoints in the database. However, they will only contain the order of the array and the user id. When the researcher clicks on the button to add the blank space, there is a creation of a new object that contains the order and the user id. This object is added to the array of touchpoints of that particular user at the correct index. Afterwards, we add a new element to the touchpoints collection inside the project document containing the user id and the order. The whitespaces can also be deleted. To achieve this, the inverse of adding a whitespace is done. We delete the object from the array of touchpoints for each user and then delete it from the database.

The last aspect from the journey map component in our project is the ability to see the detailed information regarding each touchpoint. For this, we used a Drawer component of the Mantine library. We can see the title, satisfaction, notes, location, photos, and videos from the touchpoints. To fulfill this functionality, we used a state with an object. When the researcher double clicks on a touchpoint, the information of it is passed on to the object in this state.

Visualizing in the Map

This functionality allows the researcher to visualize all the touchpoints in the map and filter them by satisfaction and by user. For this, several markers were custom made by using an SVG example as a base and then the colour was changed in the component. To use google maps, a library called google-map-react was installed and an API key in google maps console was created. Moreover, the markers take longitude and latitude, which are defined when the data is fetched at the beginning. There is also a Drawer component from Mantine when the researcher clicks the markers. In contrast to the journey maps where we have all the data from the touchpoints from the phase where the data is fetched, when a researcher clicks on a Marker, first we read the data from the database and then pass it to the state.

Overview

In the overview, the researcher can see numbers corresponding to the amount tags, the amount of users with a particular satisfaction, participants, and genders. These are shown in tables which have customized horizontal and vertical axis. The process in this case is having predefined table headers when the user selects the desired options, and subsequently, the data of the table is read from the database and passed on to a state to put in the table.

Settings of each Project

The settings of each project are divided into adding tags, visualising and deleting tags, deleting the project, and changing the status of the project.

To add tags, the researcher just writes the name and the tag is added to the tags collection in the database. The system only allows for a researcher to add up to ten tags, this comes from a limitation from Firestore library, where the *array-contains* in a query only allows for up to 10 elements in the array.

In the following section, the researcher can see all the tags and can click on the x button to delete them. To do this, the tag is deleted from the tags collection, then we get all the touchpoints and run the *arrayRemove()* function on all the touchpoints to remove the tag from that touchpoint. Finally, the state containing the tags is updated.

To delete the project, we first loop through all participants in the project and delete all data in the Firebase storage regarding each one. Then, we delete the document inside the projects collection of the participant that connects the participant to this project and get out of the loop. Afterwards, we delete the tags, touchpoints and participants collections and finally, delete the project document inside the projects collection.

In the final section of the settings, we can change the status of the project. To achieve this, the researcher selects the desired status, and the field in the document of the project is changed to that state.

This page is intentionally left blank.

Chapter 6

Software Testing

Software testing is the phase in which we check whether the developed project complies with the requirements and does not have defects. The main goal in this chapter is to guarantee users a quality product.

Testing can be divided into black box and white box testing. "Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths." [29]. "White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security." [30].

Regarding our project, we are going to focus mainly on black box testing in order to test the functionalities and try to encounter any defects in our code. The techniques that will be used are end-to-end testing for the Web application and usability testing for the PWA. White box testing was already made in the development phase, and techniques such as unit testing are particularly difficult to apply as our code has asynchronous functions that read and write from the database. These functions don't have an input and consequently they don't return any value, thus not having a measurable output. Most functions work with states and by updating the correspondent state.

6.1 End-To-end Testing

End-To-End Testing is a software testing method that "tests the functionality and performance of an application under product-like circumstances and data to replicate live settings." [41]. The goal of this method is to simulate real user scenarios from beginning to end. The tool we are going to use to ensure the implementation of this method is Selenium IDE [40]. It allows us to write tests without any code and then execute them all at once. In addition, tests can be exported to JavaScript.

The plan to fulfil this method is to look at the requirements and test them by simulating the user actions to achieve a particular one. Also, all user actions need to be simulated, such as providing an input wrong information, for example, a non registered user email or a wrong password for an already registered account. In Figure 6.1, an example of a test can be seen corresponding to a correct log-in.

	Command	Target	Value
1	✓ <i>open</i>	http://localhost:3000/	
2	✓ <i>click</i>	id=email	
3	✓ <i>type</i>	id=email	fake@mail.com
4	✓ <i>click</i>	id=password	
5	✓ <i>type</i>	id=password	123456
6	✓ <i>click</i>	id=sign-in	
7	✓ <i>execute script</i>	return window.location.href	http://localhost:3000/home

Figure 6.1: Log-in test using Selenium IDE

In the figure shown above, there is a table with three columns: command, target, and value. Command is where the type of action desired is defined; this can be an open, a click, a type, for example. Target is where the command is concluded; this can be an URL, an id, for example. Value is an input for the target or a the return value of a function defined in the target, this can be any type of text. In this particular case, the test simulates that the user should open the website, then click on the field to enter the email, type the email, click on the field to enter the password and then type it. Lastly, the test simulates that the user should click on the sign-in button and the page where the URL with the path defined as /home should load. The green ticks show whether the test was successful. If it failed at some stage, it would appear as a red tick at that specific stage. The full list of the tested used cases can be seen in Table 1 in Appendix B. Additionally, the following tables in Appendix B show the description of the various tests made.

To achieve the tests shown successfully, some modifications to the code had to be made, such as adding identifiers to inputs and buttons to allow Selenium to identify them. All of these tests passed, some not on the first attempt, but it implied only small modifications like sanitising error messages, that is, changing the texts in the alert prompts due to the firebase error messages being too large for the alert prompt too handle. As can be seen in Table 1 from Appendix B, some tests had to be done by hand due to being difficult and time consuming to do with automation because of functionalities like drag-and-drop and some ids being hard to reach. This is largely because of the use of libraries like Mantine for the front-end. When some identifier was passed to a Mantine component, Selenium could not find it, or it took too much time, i.e. more than 30 seconds, to find it.

6.2 Usability Testing

After the author of this thesis had fully developed the mobile application, some tests were performed to ensure that the functionalities were indeed working. After this stage, we are ready to let some real users try the application and give their opinion on whether the application is usable or not. In addition, with the real user's opinion, more improvements can be made with the conclusions taken.

6.2.1 Participants

The participants in this test are of a variety of ages. The range is between twenty-two and fifty-nine years old. Three people of this test are used to working with software development, are young and have knowledge in using mobile applications. Two users are older and they only use mobile applications for messages and social media. In this way, we can see how different users with different experience in technologies can reach the proposed goals. There was a decision to make the target audience any user of services.

By doing some research, the ideal number of users should be five to "get close to user testing's maximum benefit-cost ratio." [34]. With this in mind, we are going to use five people for usability testing of our mobile application.

6.2.2 Procedure

In usability testing, the test should be performed in a calm environment without distractions in order to the user understand the tasks to be attained. To start the test, the objective of it and the project was explained to the user. Afterwards, the user signed a consent form saying that test is voluntary and will be used to improve the application by the information provided. The installation of the application was guided by the author and it will not count to the test.

The test procedure consists of nine tasks:

1. Register in the application;
2. Enter the project with the provided QR Code;
3. Submit a new touchpoint in the project with random information, but adding at least a photo, a video and GPS location;
4. Edit the submitted touchpoint title to "This is a test";
5. Turn off all the internet sources from the mobile device and close the application;
6. Re-do step 3 with the internet connection turned off;
7. Turn on internet connection again and check if the touchpoint is being saved online;
8. Change the password for your account;
9. Logout from the application;

For each of the tasks, the author took notes on how many clicks the user took to reach that specific task. After the procedure was completed, the users responded to some questions and wrote some conclusions on what can be improved.

6.2.3 Results

The results of this test were measured by the number of clicks to reach a specific task. In Table 6.1, the results can be seen for each user.

Users \ Tasks	Tasks								
	1	2	3	4	5	6	7	8	9
User 1	11	3	14	4	4	14	4	7	1
User 2	10	5	12	3	3	13	2	6	2
User 3	10	3	12	3	3	13	4	5	1
User 4	10	3	13	4	3	12	2	5	1
User 5	15	3	12	4	4	13	2	5	1

Table 6.1: Metrics for each user for each completed task

In Table 6.2, the expected number of clicks for each task can be seen. This number helps us to draw some conclusions about the results of the tests. Task 4 can have 3 or 4 clicks due to having only Wi-Fi or having both Wi-Fi and mobile data on.

Tasks	1	2	3	4	5	6	7	8	9
Expected number of Clicks	10	3	12	3 or 4	3	12	2	5	1

Table 6.2: Expected Number of Clicks for Each Task

In Table 6.3, we can see what is the mean number of clicks for each task. By having this table, we can see which task has some kind of big discrepancy from the expected number of clicks in Table 6.2. For a big discrepancy, we are going to consider three clicks of difference. Note that the mean shown in this table is a rounded number.

Tasks	1	2	3	4	5	6	7	8	9
Mean number of Clicks	11	3	13	4	3	13	3	6	1

Table 6.3: Mean Number of Clicks for Each Task

As the tables above show, the test achieved quality results, all tasks were completed and the discrepancy is not very high between the mean number of clicks and the expected number of clicks. The tasks that differ are 1, 3, 6, 7 and 8.

Regarding task 1, this differs from the expected number because user 5 had some extra clicks on inserting the date of birth. For the rest of the users results are normal and the behaviour was as expected.

Regarding task 3, the behaviour was also as expected. However, user 1 has some extra clicks due to wanting to change the notes and the title before submitting. This is a very large task, but the test results show that the screen is well designed and intuitive. This analogy also applies to task 6 which has the same screen.

Regarding task 7, some users showed difficulty finding in the application the screen where the touchpoints submitted offline were syncing to the online database.

Regarding task 8, user 1 made a mistake when inserting his current password and two extra clicks were necessary to achieve this task.

Questionnaire Answers

The questionnaire had some questions which were answered in a linear scale from 0 to 10 and a question with a yes, no and maybe. The first question is whether the application is easy to use. Figure 6.2 shows the results for this question.

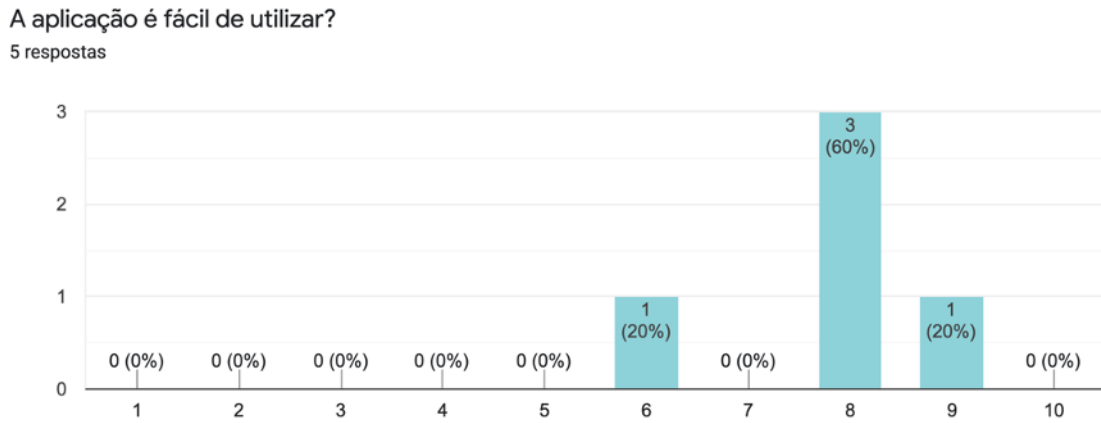


Figure 6.2: Question one

As can be seen, the majority of users chose option 8, which means that we made an application easy to use, achieving one of our goals.

The second question asks the user if it was simple to enter a project. Figure 6.3 shows the results for this question.

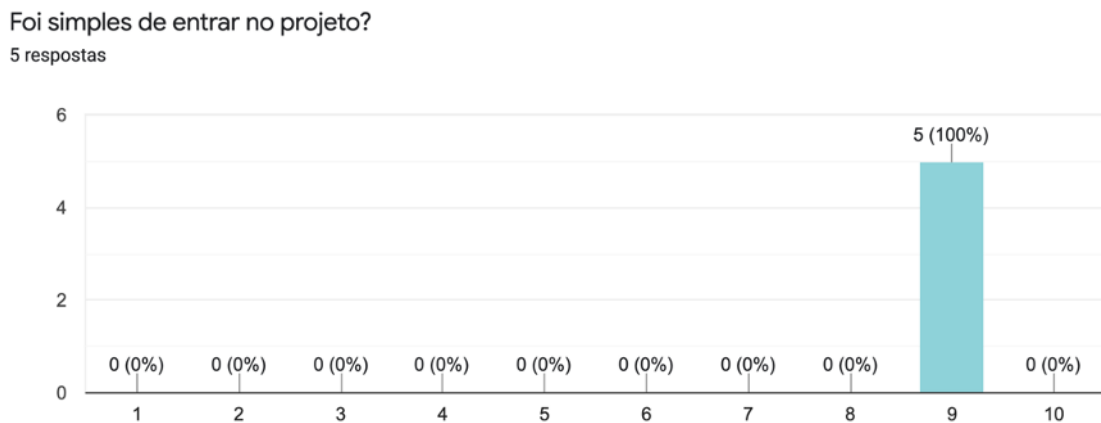


Figure 6.3: Question two

Option number 9 was chosen by all users. Having this number signifies that users liked the functionality; therefore, it can be concluded that it is very straightforward to enter a new project.

The third question asks the user if he would use the application in a real-life scenario. Figure 6.4 shows the results.

Se a aplicação fosse utilizada num hotel para avaliar o serviço, utilizava-a?
5 respostas

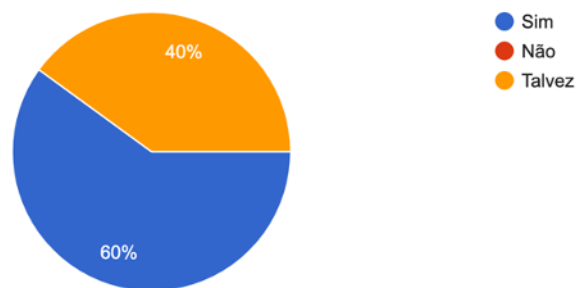


Figure 6.4: Question three

The pie chart shows that 60% of the users said yes and 40% maybe. By not having any answer with no, it can be concluded that users would use it in a real-life scenario.

The fourth question asks the user if he feels the application design is appealing. Figure 6.5, shows the results.

O design da aplicação é apelativo?
5 respostas

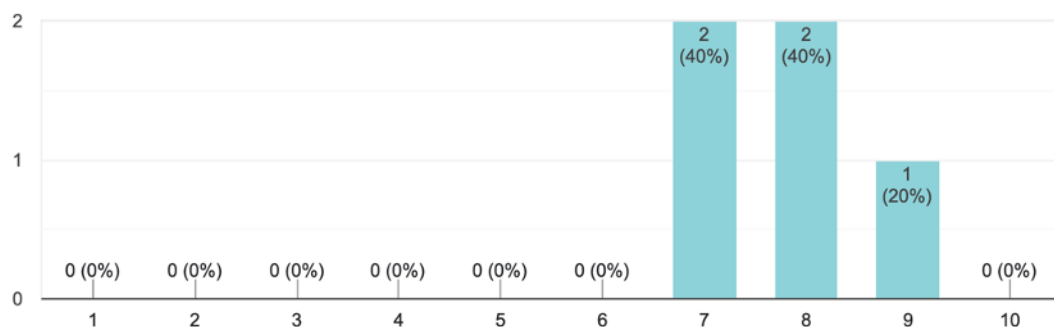


Figure 6.5: Question four

By only having high numbers, 7, 8 and 9, we can conclude that the design is appealing, therefore achieving another one of our goals which states that the User Interface needs to be intuitive and straightforward.

The answers to the last question asking if the application worked well without internet connection can be seen in Figure 6.6.

Achou que a aplicação funciona bem sem internet?

5 respostas

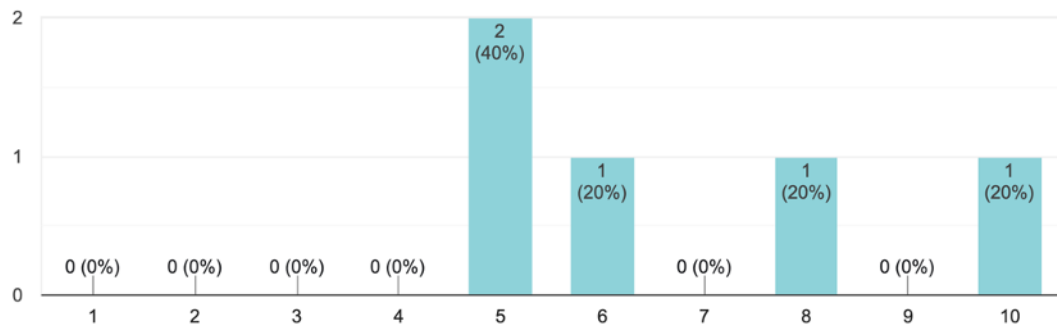


Figure 6.6: Question five

It can be seen that the functionality of not having internet has worked for some users but others did not understand how the application worked without having an internet connection. Therefore, there is room for improvement in this area.

Overall, all the functionalities worked in the desired manner except for the deferred synchronisation, as users did not understand how it worked. They completed all the tasks, which meant that the test was completed with success.

Sugestions

After all responses, users had the opportunity to write their own suggestions. Only one user did not have any suggestions. The suggestions were the following:

- Getting into the project is easy, but the page is a bit confusing, having a place to put the token and another to scan the QR code.
- Greater user interaction through more suggestive warning messages whenever an action was taken or an error occurred.
- Improve the colour scheme; black is too dark; maybe a dark grey will achieve a similar effect but a bit more appealing. Regarding the functioning of the platform, I have no suggestions.
- The application took a long time to save the submitted touchpoint offline, but otherwise it worked fine.

6.2.4 Conclusions

To a large extent, the tests were successful due to users completing all tasks with a low degree of difficulty. The only problem was with task 7 where users did not know where to find the screen for the syncing part, but having gone back to the main screen of the application was then obvious for them.

All the participants of this test perceived the application objective and the majority would later use it in a real-life scenario. However, by the user suggestions we can still have room for improvement regarding the user interface in terms of color and there is also some confusion in some inputs. In addition, more alerts are needed to warn users of their actions or if errors occurred.

Lastly, we can say that our application follows a good path in terms of usability despite having some suggestions to improve it. For future development into a more robust version, some of these suggestions can be taken into account and be used to improve the application.

6.3 Software Testing Conclusions

By doing End-To-End and Usability testing, an evaluation of the developed products can be performed. Overall, both products achieved what was expected. All functionalities worked as intended and some even exceeded the testers' expectations, for example, reading the QR code for entering a project. Only one functionality, deferred syncing, was not accepted as expected. However, the functionality worked in the correct format. The problem here is that the approach to this solution was not the best. There are ways to correct this problem. The implication of this problem is that the users can not leave the screen while the touchpoints are syncing. Other testers did have more suggestions, more suggestive warning messages, however, they did not state clearly what they wanted. So, in order to achieve this suggestion, when an action is taken, the application should respond in any way and the texts could be improved. The implications of this problem are based on the user being lost in their actions in the application.

With these tests, we can conclude that there is no need for changes in this iteration of the platform. In a future version, some design changes can be made, mainly changing the colour scheme and changing the way deferred syncing is done. Instead of doing it in a component, the optimal solution would be to do it in the service worker. Also, there is a need to perform more tests in the mobile application with respect to warning messages. This is for having a sense of the users' opinions in relation to the warning messages and do the adequate changes.

Chapter 7

Conclusion

The theme of this thesis was proposed by Department of Informatics Engineering (DEI) and the main objective was to develop a tool for evaluating services. To achieve this, a Progressive Web Application (PWA) and a Website were developed. With the PWA, feedback is collected from service users using mobile ethnography as a data collection method. The Website uses service design guidelines to be able to evaluate the service fairly. The data being evaluated is collected from the PWA. The development followed a scrum-based methodology. In the first semester, a study was carried out on the subject of this thesis, this included mobile ethnography, service design and a search for similar tools to the one developed. There was also a study on what technologies should be applied to it, there was a gathering of requirements and tasks where planned according to the time available for the whole project. In the second semester, there was a refinement of the architecture and a focus on development and testing.

The first step taken in this project was to study services and what they represent. This serves as context on what was developed. Subsequently, there was a study on Service Design that helped later in the requirements for the Website. Then, there was a study on Mobile Ethnography to gain more knowledge to apply it later to the requirements of the PWA. Lastly, there was some research to find some similar tools in the market to get a sense of what will be needed for the functional requirements and connect them to the research done of Mobile Ethnography and Service Design.

The second step was to study the technologies. As the author's knowledge with the React framework was already good, this was chosen to develop both the PWA and the web application. There was also a study for other technologies to gather information and see if it was beneficial despite the time spent learning a new technology. The conclusion for this problem is that is not beneficial and the other technologies have greater learning curves. For the back-end, Firebase was the chosen technology. This is related to the fact that integrating it with React is a simple process and by having a No-SQL document-oriented database, it has good scalability. Also, it has more services like authentication and a storage without needing a physical server.

Following this study, a document was prepared with the requirements specification. To achieve the requirements, there was a study done in the state-of-the-art in which similar tools to the one developed were found and helped gathering the requirements. There was a focus on use cases to represent functional requirements. These contain some relevant information on how a user will reach this functionality and the frequency of use. The MoSCoW method was also applied to functional requirements to prioritize them. The non-functional requirements were gathered by having discussions with the advisor of this

thesis and decided which were the most adequate for the tool. After having all the requirements specified, there was a need to design some mock-ups. First, an User Experience Diagram (UED) for both the PWA and web application was done in order to assign the requirements to each screen and how the navigation would work, making it a coherent design. Then, using a tool called inVision, the mock-ups were designed to help save time in the development phase.

In order to draw solutions from the requirements to our software system, we have implemented a software architecture. For this, the C4 model was used. It also helped in the development phase, allowing to see the how the back-end connects to the front-end and dividing the several screens into components.

The development stage started with the creation of the project files and the set up of the development and staging environments. As development progressed, the author tested each functionality before developing the next one, this was also an attempt to minimize bugs in the staging environment. All the "Must Have" and "Should Have" requirements were implemented. One of the requirements "Could Have" was not, due to not being strictly necessary for this project as there is no need to export data in this first version of the product. This makes this project achieve the threshold of success as it met all the minimum conditions.

After the development was completed, we proceeded to testing to evaluate whether the functionalities are indeed well developed or not. The first method was end-to-end testing, using both automated and manual testing. This was a success, as all tests passed with minimum changes. These changes were changing some ids to inputs and button in order for Selenium to read them. The second method was usability testing. Some suggestions were made and all users performed the tasks presented in an efficient manner. This usability testing made it clear that the non-functional requirement usability with the metric of having a simple to use User Interface (UI) is fulfilled.

The next step was set up the development environment. For this stage, both applications front-end were served on DEI internal servers. For the back-end, it is served from google's servers. The front-end server for the PWA needed to be served with https. To achieve this goal, let's encrypt certbot was used. Also, both servers are serving the front-end using nginx.

Several challenges occurred during the development of this project. One of them was managing the time needed for developing two different products, the PWA and the web application. The focus of development could not be on one product at a time, as to test some functionalities, they needed to be concluded on the web application as well as on the PWA. So, going back and forth between the two applications code could be complicated sometimes, due to losing the focus on what was being developed and therefore hinder the efficiency of development and being more prone to make mistakes. Another challenge was configuring the API keys for Google maps and integrating them into our code in the PWA, as they would only work after the project was built. This made testing in the development environment more laborious.

For future progress of the tool, some testers suggested better offline syncing, as they did not understand how that functionality worked. The solution to offline syncing was doing the syncing part in the initial screen, this is not the optimal solution as the user can not leave that screen until all is synced. In the future, it could be done on the service worker for better navigation and a more seamless experience. This way, users could use the app while syncing. Another improvement could be made to the web application, such as loading the content of a page. The current version is missing some feedback in that particular part.

The solution is to show more feedback to end users by using spinners while content is loading. Also, the PWA does not show the archived and finalized projects. In the next version, showing them can be a good improvement.

Doing this thesis was a rewarding experience due to putting into practice software engineering skills thought during the degree. It was also a chance to gain more experience on the various steps of a software engineering project while performing mobile and web development.

References

- [1] 10 platforms for mobile ethnography. <https://www.insightplatforms.com/10-platforms-for-mobile-ethnography/>. Accessed on October 2021.
- [2] Amazon. Aws calculator. <https://calculator.aws/>. Accessed on December 2021.
- [3] Atlassian. react-beautiful-dnd library. <https://github.com/atlassian/react-beautiful-dnd>. Accessed on November 2021.
- [4] Several Authors. This is service design doing. <https://www.thisisservicedesigndoing.com>. Accessed on October 2021.
- [5] Several Authors. This is service design thinking. <http://thisisservicedesignthinking.com>. Accessed on October 2021.
- [6] The World Bank. Services, value added (% of gdp). <https://data.worldbank.org/indicator/NV.SRV.TOTL.ZS?end=2020&start=1995&view=chart>. Accessed on October 2021.
- [7] The World Bank. Services, value added (% of gdp) - united states. <https://data.worldbank.org/indicator/NV.SRV.TOTL.ZS?locations=US>. November on October 2021.
- [8] John D. Brewer. *Ethnography*. Open University Press, 2000.
- [9] Simon Brown. C4 model. <https://c4model.com/>. Accessed on March 2022.
- [10] React Native Community. React native net info. <https://github.com/react-native-netinfo/react-native-netinfo>. Accessed on May 2022.
- [11] Danny Connell. Localbase. <https://github.com/dannyconnell/localbase>. Accessed on May 2022.
- [12] Brian Dean. Netflix subscriber and growth statistics: How many people watch netflix in 2021? <https://backlinko.com/netflix-users>. Accessed on October 2021.
- [13] WordPress Engine. jquery versions in wordpress. <https://wpengine.com/support/including-a-different-jquery-version-in-wordpress>. Accessed on November 2021.
- [14] Expo. Expo sdk. <https://expo.dev/>. Accessed on October 2021.
- [15] Expo, Software Mansion, and Callstack. React navigation. <https://reactnavigation.org/>. Accessed on May 2022.
- [16] Facebook. React.js github. <https://github.com/facebook/react>. Accessed on October 2021.

-
- [17] Facebook. React.js jsx. <https://reactjs.org/docs/introducing-jsx.html>. Accessed on October 2021.
- [18] Facebook. React.js website. <https://reactjs.org/>. Accessed on November 2021.
- [19] Facebook. Redux. <https://redux.js.org/>. Accessed on October 2021.
- [20] Jason Fernando. What is gross domestic product (gdp)? <https://www.investopedia.com/terms/g/gdp.asp>. Accessed on June 2022.
- [21] OpenJS Foundation. What is jquery? <https://jquery.com/>. Accessed on October 2021.
- [22] Google. Firebase authentication. <https://firebase.google.com/docs/auth>. Accessed on October 2021.
- [23] Google. Firebase cloud firestore. <https://firebase.google.com/docs/firestore>. Accessed on October 2021.
- [24] Google. Firebase cloud functions. <https://firebase.google.com/docs/functions>. Accessed on October 2021.
- [25] Google. Firebase hosting. <https://firebase.google.com/docs/hosting>. Accessed on October 2021.
- [26] Google. Firebase storage. <https://firebase.google.com/docs/storage>. Accessed on October 2021.
- [27] Google. google-map-react library. <https://www.npmjs.com/package/google-map-react>. Accessed on November 2021.
- [28] Google. What is angular? <https://angular.io/guide/what-is-angular>. Accessed on October 2021.
- [29] Thomas Hamilton. What is black box testing? techniques, example types. <https://www.guru99.com/black-box-testing.html>. Accessed on June 2022.
- [30] Thomas Hamilton. What is white box testing? techniques, example types. <https://www.guru99.com/white-box-testing.html>. Accessed on June 2022.
- [31] Mantine. Mantine library. <https://mantine.dev/>. Accessed on May 2022.
- [32] Birgit Muskat, Matthias Muskat, and Anita Zehrer. Qualitative interpretive mobile ethnography. *Anatolia*, 2017.
- [33] NTask. Scrum artifacts | what are they how to incorporate them in agile framework? <https://www.ntaskmanager.com/blog/scrum-artifacts/>. Accessed on November 2021.
- [34] Jakob Nielsen on Nielsen Norman Group. How many test users in a usability study? <https://www.nngroup.com/articles/how-many-test-users/>. Accessed on June 2022.
- [35] O'Reilly. Pwa technical requirements. <https://www.oreilly.com/library/view/progressive-web-application/9781787125421/e89ac7bb-8251-41ed-ad3f-6ed793a73b9d.xhtml>. Accessed on May 2022.
- [36] Oxford. Definition of ethnography. <https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095759601>. Accessed on October 2021.

- [37] React Router. Browserrouter. <https://reactrouter.com/web/api/BrowserRouter>. Accessed on October 2021.
- [38] React Router. Hashrouter. <https://reactrouter.com/web/api/HashRouter>. Accessed on October 2021.
- [39] Fabian Segelström, Bas Raijmakers, and Stefan Holmlid. Thinking and doing ethnography in service design. *IASDR: Rigor and Relevance in Design, 1–10*, 2009.
- [40] Selenium. Selenium ide. <https://www.selenium.dev/selenium-ide/>. Accessed on June 2022.
- [41] SmartBear. Combine api and ui testing for confidence at every layer of your application. <https://smartbear.com/solutions/end-to-end-testing/>. Accessed on June 2022.
- [42] Apoorva Srivastava, Sukriti Bhardwaj, and Shipra Saraswat. Scrum model for agile methodology. *2017 International Conference on Computing, Communication and Automation (ICCCA)*.
- [43] StackOverflow. 2020 developer survey. <https://insights.stackoverflow.com/survey/2020>. Accessed on October 2021.
- [44] Statista. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021. <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>. Accessed on November 2021.
- [45] Statista. Total size of the public cloud computing market from 2008 to 2020. <https://www.statista.com/statistics/510350/worldwide-public-cloud-computing/>. Accessed on November 2021.
- [46] Stripe. Stripe official | payment processing platform for the internet. <https://stripe.com/en-pt>. Accessed on June 2022.
- [47] Ash Turner. How many smartphones are in the world? <https://www.bankmycell.com/blog/how-many-phones-are-in-the-world>. Accessed on October 2021.
- [48] Wikipedia. Mobile ethnography. https://en.wikipedia.org/wiki/Mobile_ethnography. Accessed on August 2022.
- [49] Wikipedia. React (javascript library). [https://en.wikipedia.org/wiki/React_\(JavaScript_library\)](https://en.wikipedia.org/wiki/React_(JavaScript_library)). Accessed on October 2021.
- [50] Wikipedia. User experience. https://en.wikipedia.org/wiki/User_experience.

Appendices

This page is intentionally left blank.

Appendix A - Use Cases

Researcher Sign Up

A researcher was told about the application by a friend and he wants to sign up on it.

Name	Researcher Sign Up
ID	02
Primary Actor	Researcher not registered on the System
Level	Blue
Description	The researcher makes an account on the system, inserting his personal data
Pre-condition	The researcher is not registered
Basic path	1: The researcher goes to the system's website 2: The researcher selects the Sign Up option 3: The researcher inserts the personal data 4: The participant accepts the legal terms and conditions 5: The Participant selects the Sign Up option 6: The System verifies the data 7: The System validates the data 8: The System adds the new researcher
Alternative path	6.a: The System finds an error in the researcher data 6.b: The System informs the researcher about the error (email already exists or password too weak) 7.a: The system notifies to change the personal data
Post-condition	Researcher is registered and he is redirected to the Dashboard page
Frequency	Medium

Table 1: Researcher Sign Up Use Case Table

Participant Sign In

A participant wants to sign in.

Name	Participant Sign In
ID	03
Primary Actor	Participant
Level	Blue
Description	The participant signs in inserting his personal data
Pre-condition	The participant is already registered on the system
Basic path	<ol style="list-style-type: none"> 1: The participant opens the application 2: The participant selects the Sign In option 3: The participant inserts the personal data (email and password) 4: The participant selects the Sign In option 5: The System verifies the data
Alternative path	<ol style="list-style-type: none"> 5.a: The System finds an error in the user data (email doesn't exist or wrong password) 5.b: The System informs the user about the error
Post-condition	Participant is signed in and he is redirected to the Choose Project page
Frequency	Medium

Table 2: Participant Sign In Use Case Table

Researcher Sign In

A researcher wants to sign in.

Name	Researcher Sign In
ID	04
Primary Actor	Researcher
Level	Blue
Description	The researcher signs in inserting his personal data
Pre-condition	The researcher is already registered on the system
Basic path	1: The researcher goes to the system's website 3: The researcher inserts the personal data (email and password) 4: The researcher selects the Sign In option 5: The System verifies the data
Alternative path	5.a: The System finds an error in the user data (email doesn't exist or wrong password) 5.b: The System informs the user about the error
Post-condition	Researcher is logged in and he is redirected to the Choose Project page
Frequency	Medium

Table 3: Researcher Sign In Use Case Table

Join a New Project

A participant enters a flight and sees the QR code to enter the airline's project. Now, he wants to join this project to give his insights to the airline.

Name	Join a New Project
ID	05
Primary Actor	Participant
Level	Blue
Description	The participant joins a new project
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the Add Project option 3: The participant scans the QR Code 4: The participant selects the Add option 5: The System inserts the participant in that project
Alternative path	3.a: The participant enters the token for the project
Post-condition	Participant is redirected to the Choose Projects page
Frequency	Medium

Table 4: Join a New Project Use Case Table

View Projects that you are Involved In

A participant wants to see the projects that he has entered.

Name	View Projects that you are Involved In
ID	06
Primary Actor	Participant
Level	Blue
Description	The participant wants to see the projects that he has entered
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant sees all the projects he has previous entered
Post-condition	Participant sees all projects that he is involved in
Frequency	High

Table 5: Views Project that you are Involved In Use Case Table

Change Password

A participant wants to change his password.

Name	Change Password
ID	07
Primary Actor	Participant
Level	Blue
Description	The participant wants to change his password
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant inserts his current password 4: The participant inserts his new password 5: The participant inserts his new password for confirmation 6: The participant selects the change password option 7: The system changes the participant's password
Alternative path	7.a: The system informs the participant to choose another password (inserted password is too weak)
Post-condition	Participant's password is changed
Frequency	Low

Table 6: Change Password Use Case Table

Delete Account

A participant wants to delete his account.

Name	Delete Account
ID	08
Primary Actor	Participant
Level	Blue
Description	The participant wants to delete his account
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant selects the delete account option 4: The participant inserts his password 5: The system informs the participant about the successful deletion
Alternative path	5.a: The system informs the participant that his password is wrong
Post-condition	Participant's account is deleted
Frequency	Low

Table 7: Delete Account Use Case Table

View all touchpoint points submitted

A participant wants to see all the touchpoints submitted for a particular project.

Name	View All Touchpoints Submitted
ID	09
Primary Actor	Participant
Level	Blue
Description	The participant wants to see all the touchpoints submitted for a project
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project he wants to see the touchpoints 3: The participant views all touchpoints submitted
Alternative path	None
Post-condition	Participant is on the project page and views all the touchpoints he submitted for a particular project
Frequency	High

Table 8: View all touchpoint points submitted Use Case Table

Edit touchpoint

A participant wants to change the photo for a specific touchpoint.

Name	Edit Touchpoint
ID	10
Primary Actor	Participant
Level	Blue
Description	The participant wants to change the photo for a touchpoint
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project we wants to change the touchpoint 3: The participant selects the specific touchpoint 4: The participant selects the add photo option 5: The participant selects the new photo in his device 6: The participant selects the save option
Alternative path	None
Post-condition	Touchpoint is modified with success and the participant is redirected to the choose touchpoints page
Frequency	Low

Table 9: Edit Touchpoint Use Case Table

Add New Touchpoint

A participant wants to add a new touchpoint.

Name	Add New Touchpoint
ID	11
Primary Actor	Participant
Level	Blue
Description	The participant wants to add a new touchpoint
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project we wants to add the touchpoint 3: The participant selects the new touchpoint option 4: The participant enters the title 5: The participant selects the satisfaction 6: The participant writes a small description in the notes section 7: The participant selects the save option
Alternative path	6.a: The participant selects the Add Photo Option 6.b: The participant selects the photo in his device 6.1.a: The participant selects the Add Video Option 6.1.b: The participant selects the video in his device 6.2: The participant selects the option for allowing the geolocation position
Post-condition	Touchpoint is submitted and the participant is redirected to the choose touchpoints page
Frequency	Normal

Table 10: Edit Touchpoint Use Case Table

Leave Project

A participant wants to leave a project.

Name	Leave Project
ID	12
Primary Actor	Participant
Level	Blue
Description	The participant wants to leave a project
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project we wants to leave 3: The participant selects the leave option
Alternative path	None
Post-condition	Participant is removed from the project and is redirected to the choose project page
Frequency	Low

Table 11: Leave Project Use Case Table

Participant's Logout

A participant wants to log out of the application.

Name	Participant's Logout
ID	13
Primary Actor	Participant
Level	Blue
Description	The participant wants to logout from the application
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant selects the logout option
Alternative path	None
Post-condition	Participant is signed out of the application and is redirected to the homescreen page
Frequency	Low

Table 12: Participant's Logout Use Case Table

Change Password

A researcher wants to change his password.

Name	Change Password
ID	14
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to change his password
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher inserts his current password 3: The researcher inserts his current password 4: The researcher inserts his new password 5: The researcher inserts his new password for confirmation 6: The researcher selects the change password option 7: The system changes the researcher's password
Alternative path	7.a: The system informs the participant to choose another password (inserted password is too weak)
Post-condition	Researcher's password is changed
Frequency	Low

Table 13: Change Password Use Case Table

Delete Account

A researcher wants to delete his account.

Name	Delete Account
ID	15
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to delete his account
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the delete account option 3: The researcher inserts his password 4: The system informs the researcher about the successful deletion
Alternative path	7.a: The system informs the researcher that his password is wrong
Post-condition	Researcher's account is deleted
Frequency	Low

Table 14: Delete Account Use Case Table

Create Project

A researcher wants to create a new project.

Name	Create Project
ID	16
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to create a new project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the create project option 3: The researcher inserts the name of the project and the legal terms 4: The researcher selects the create project option 5: The system creates the new project
Alternative path	None
Post-condition	Researcher is redirected to project created page and the project token and QR Code is displayed
Frequency	Normal

Table 15: Create Project Use Case Table

View Own Projects

A researcher wants to view his own projects.

Name	View Own Projects
ID	17
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to view his own projects
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option
Alternative path	None
Post-condition	Researcher is redirected to view projects page and all the projects are displayed
Frequency	High

Table 16: View Own Projects Use Case Table

Finalize Projects

A researcher wants to finalise a project to not collect more data.

Name	Finalize Projects
ID	18
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to finalize a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the finalize option of the specific project
Alternative path	None
Post-condition	Project is moved from the active to the finalized section
Frequency	Normal

Table 17: Finalize Projects Use Case Table

Archive Projects

A researcher wants to archive a project because all of the data is analysed.

Name	Archive Projects
ID	19
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to archive a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the archive option of the specific project
Alternative path	None
Post-condition	Project is moved from the finalized to the archived section
Frequency	Normal

Table 18: Finalize Projects Use Case Table

View Project Journeys

A researcher wants to analyze the project journey of a specific project.

Name	View Project Journeys
ID	20
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to analyze a project journeys
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page
Alternative path	None
Post-condition	Project journeys are displayed
Frequency	High

Table 19: View Project Journeys Use Case Table

Export Data

A researcher wants to export data from a specific project.

Name	Export Data
ID	21
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to export data from a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the export data option
Alternative path	None
Post-condition	Data is downloaded in CSV format
Frequency	Low

Table 20: Export Data Use Case Table

Manipulate Project Journeys

A researcher wants to re-organize the order of the project journeys of the several participants for comparison.

Name	Manipulate Project Journeys
ID	22
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to re-organize the order of the project journeys
Pre-condition	The researcher is already registered and signed in on the system
Basic path	<ol style="list-style-type: none">1: The researcher opens the website2: The researcher selects the view projects option3: The researcher selects the specific project4: The system redirects to the project page5: The researcher selects the data option6: The researcher selects the Add option to enter an empty space between touchpoints7: The researcher drags and drops a touchpoint to certain location8: The researcher selects the save changes option
Alternative path	None
Post-condition	The new organization of data is saved in the database
Frequency	High

Table 21: Manipulate Project Journeys Use Case Table

Delete Project

A researcher wants to delete a project.

Name	Export Data
ID	23
Primary Actor	Researcher
Level	Blue
Description	A researcher wants to delete a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Delete Project option 6: The researcher inserts his password 7: The system deletes the project
Alternative path	7.a: The password inserted his wrong
Post-condition	Project is deleted from the database
Frequency	Low

Table 22: Delete Project Use Case Table

Show Project Map

A researcher wants to see the map showing all the touchpoints.

Name	Show Project Map
ID	24
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to see the map showing all the touchpoints
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Map option
Alternative path	None
Post-condition	The researcher is redirected to the Map Page
Frequency	High

Table 23: Show Project Map Use Case Table

Show Overview Graph

A researcher wants to see the overview graph.

Name	Show Overview Graph
ID	25
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to see the overview graph
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Overview option
Alternative path	None
Post-condition	The researcher is redirected to the Overview Page
Frequency	High

Table 24: Show Overview Graph Use Case Table

Researcher's Logout

A researcher wants to logout from the website.

Name	Researcher's Logout
ID	26
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to logout from the website
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the logout option
Alternative path	None
Post-condition	Researcher is signed out of the website and is redirected to the homepage
Frequency	Normal

Table 25: Researcher's Logout Use Case Table

This page is intentionally left blank.

Appendix B - User Interface

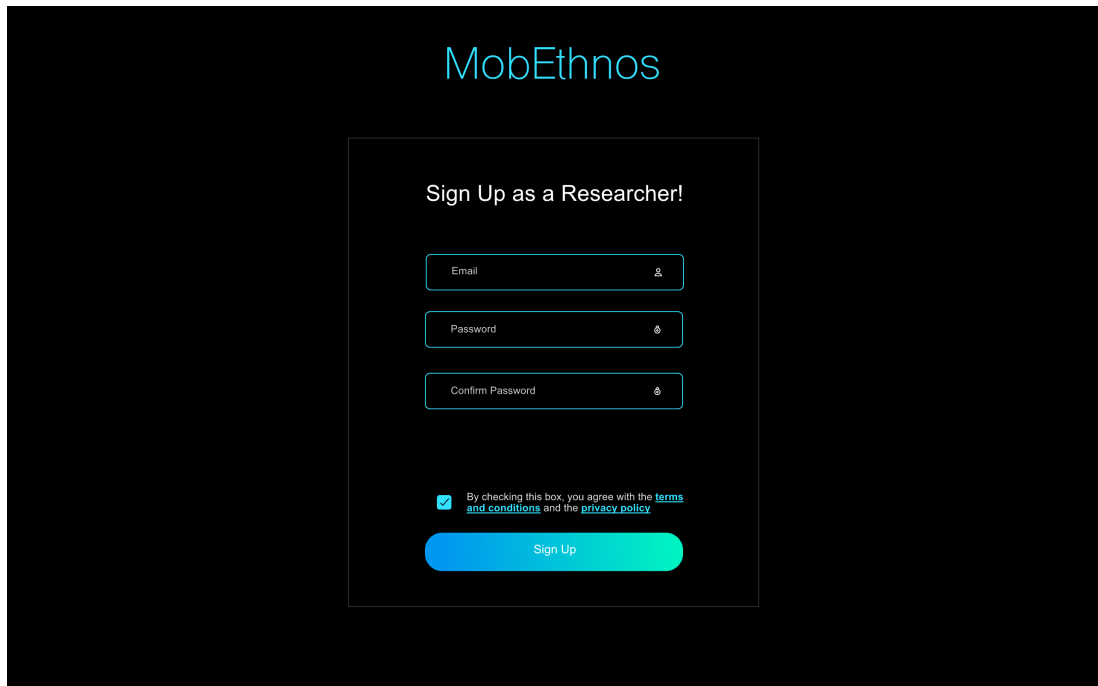


Figure 1: Sign Up User Interface

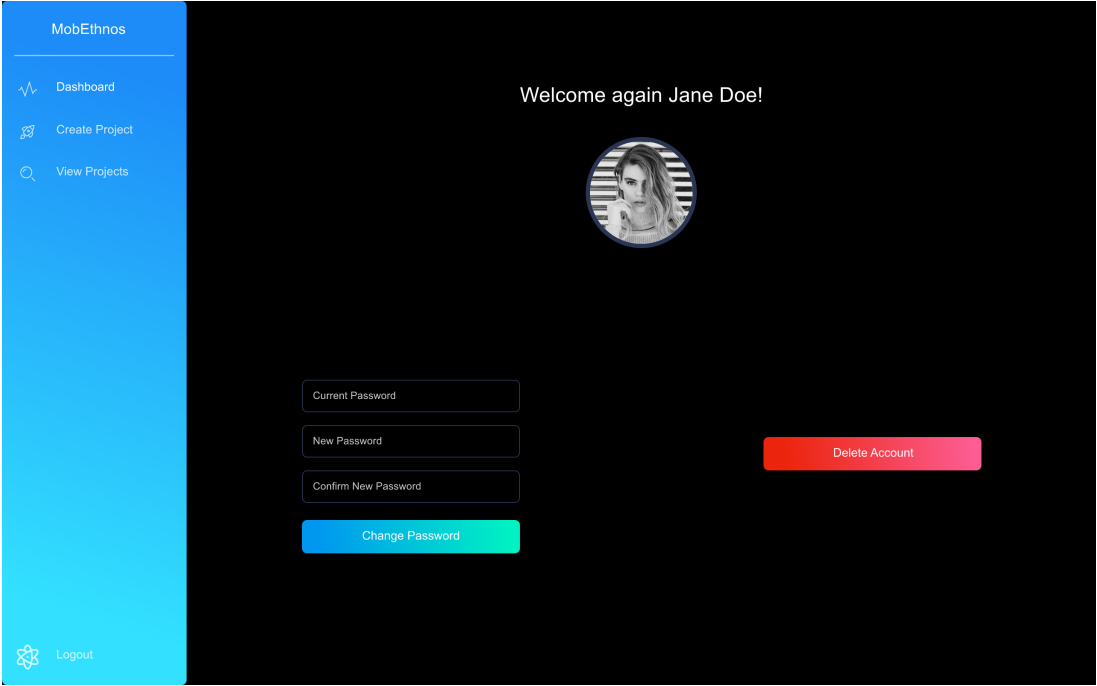


Figure 2: Dashboard User Interface

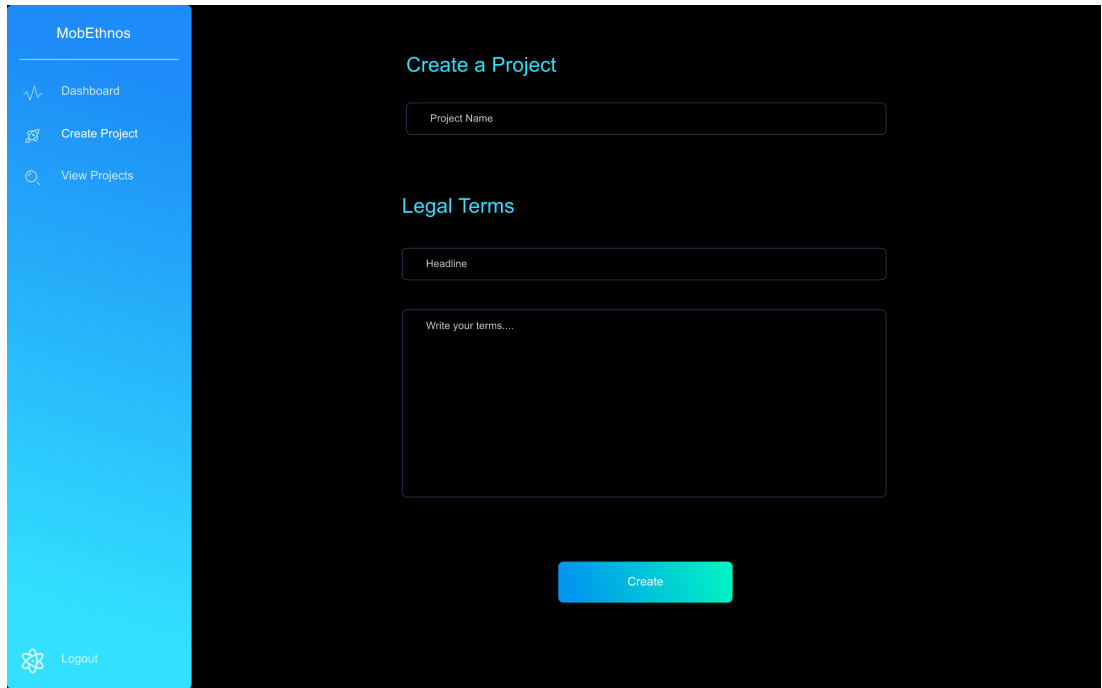


Figure 3: Create Project User Interface

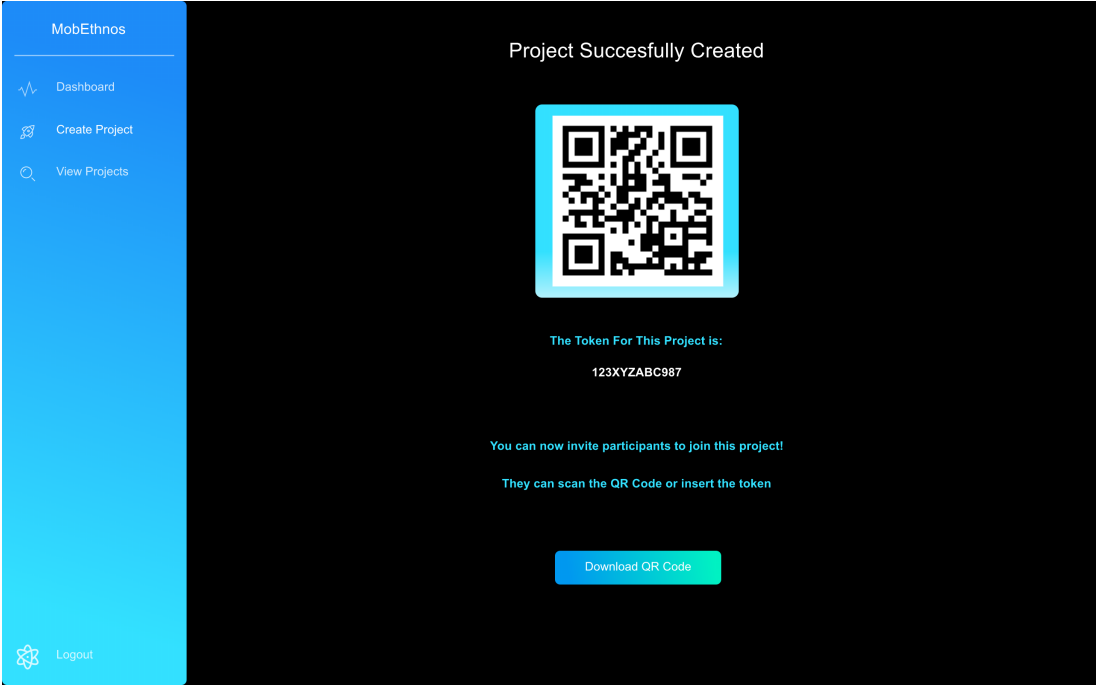


Figure 4: Project Created User Interface

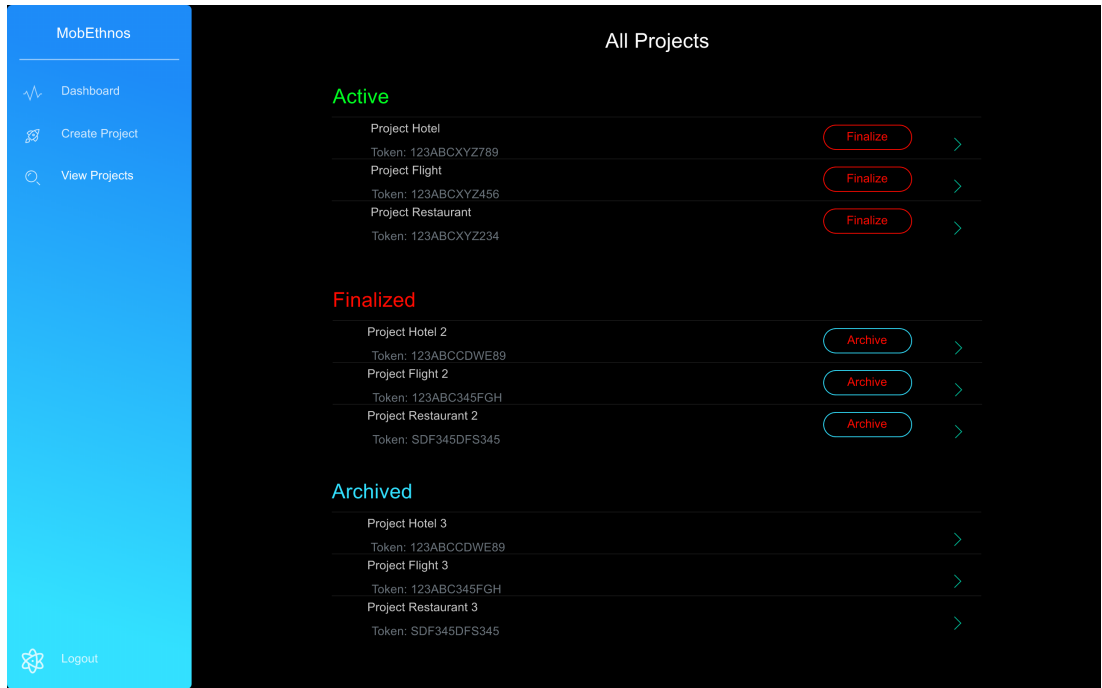


Figure 5: View Projects User Interface

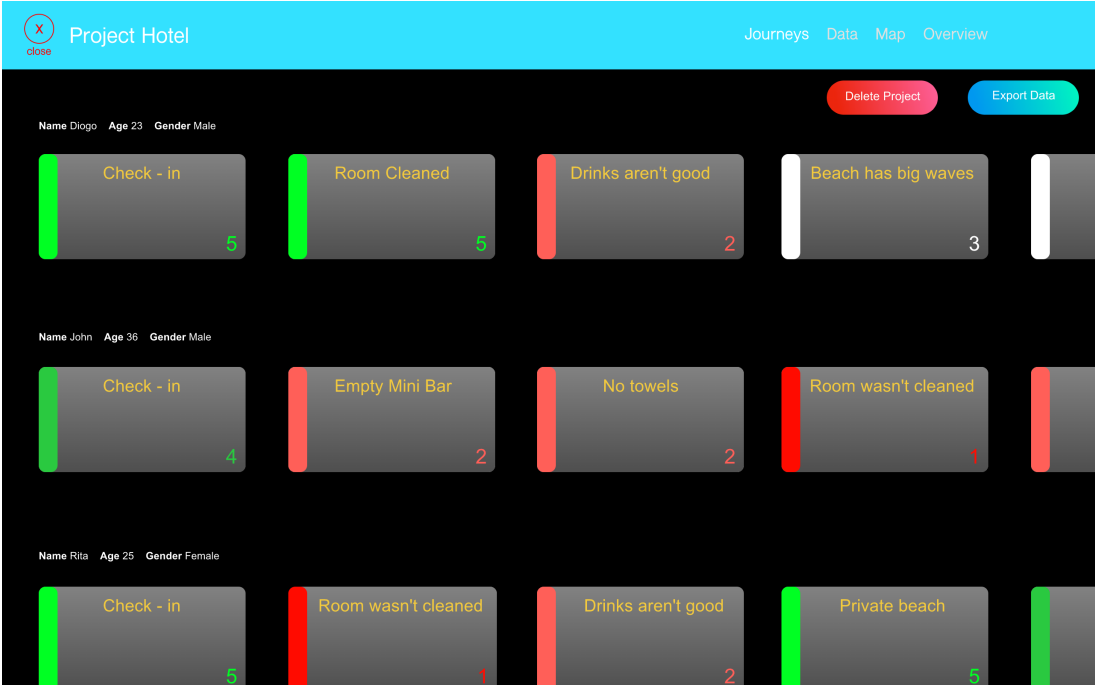


Figure 6: Project Journeys User Interface

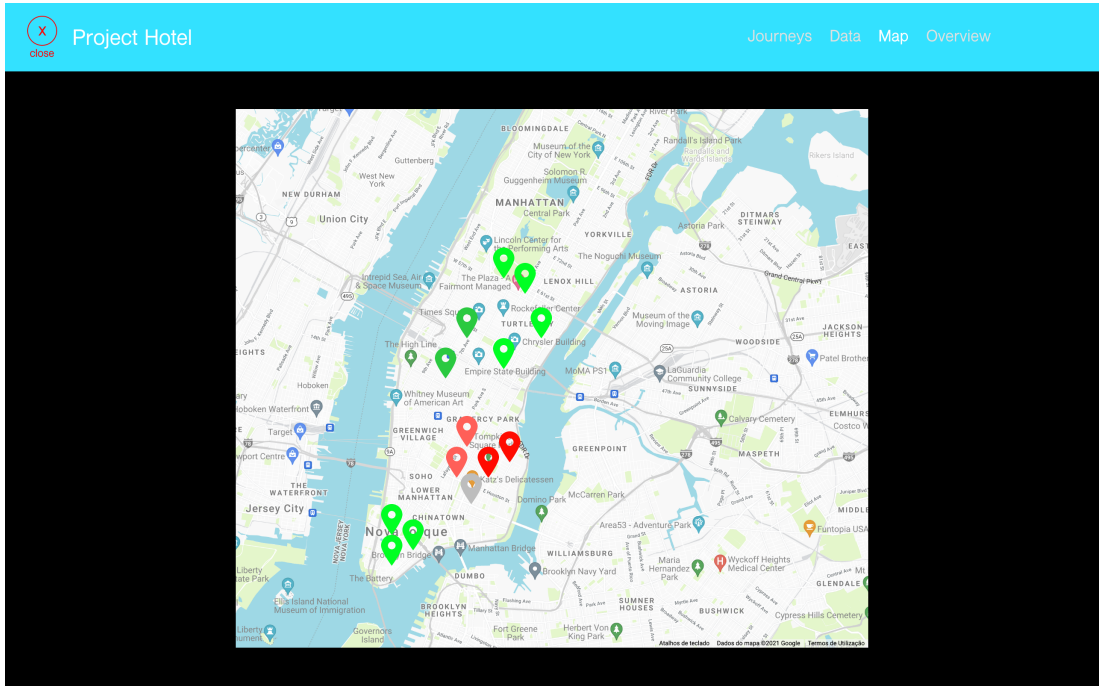


Figure 7: Project Map User Interface

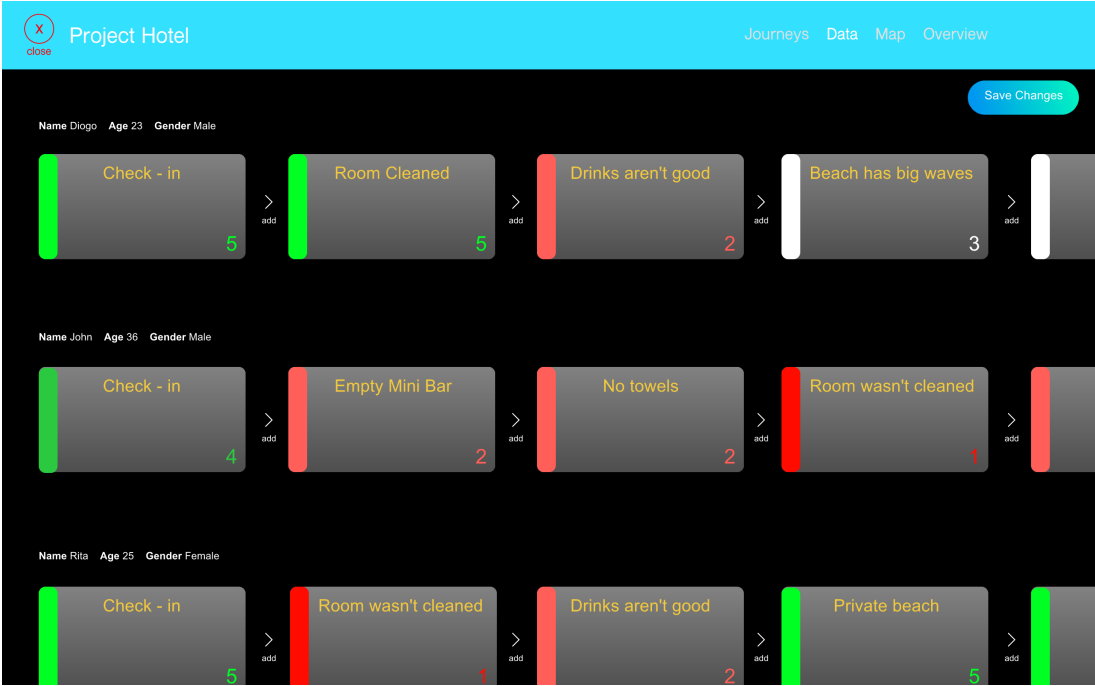


Figure 8: Project Data User Interface



Figure 9: Project Overview User Interface

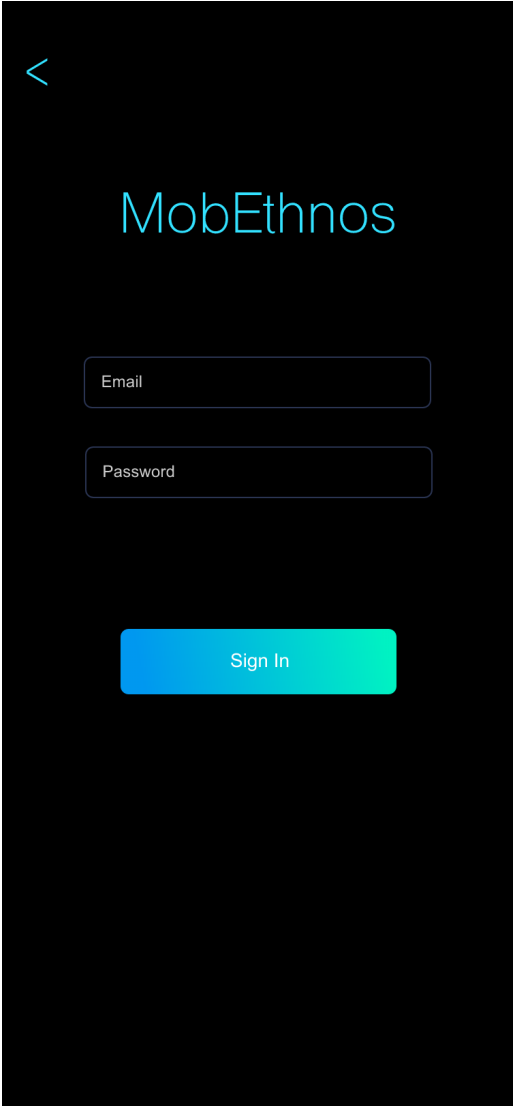


Figure 10: Sign In Interface.

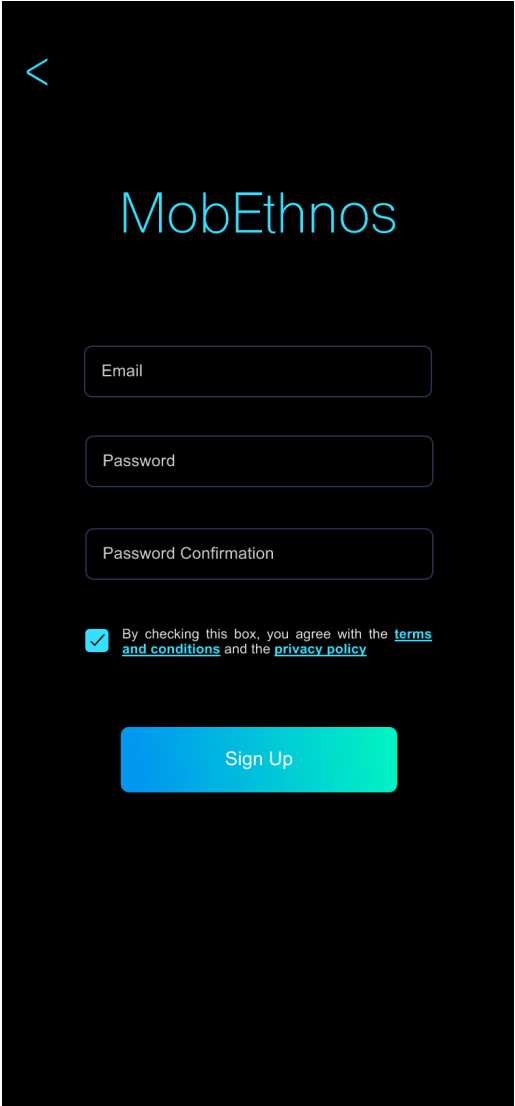


Figure 11: Sign Up User Interface.

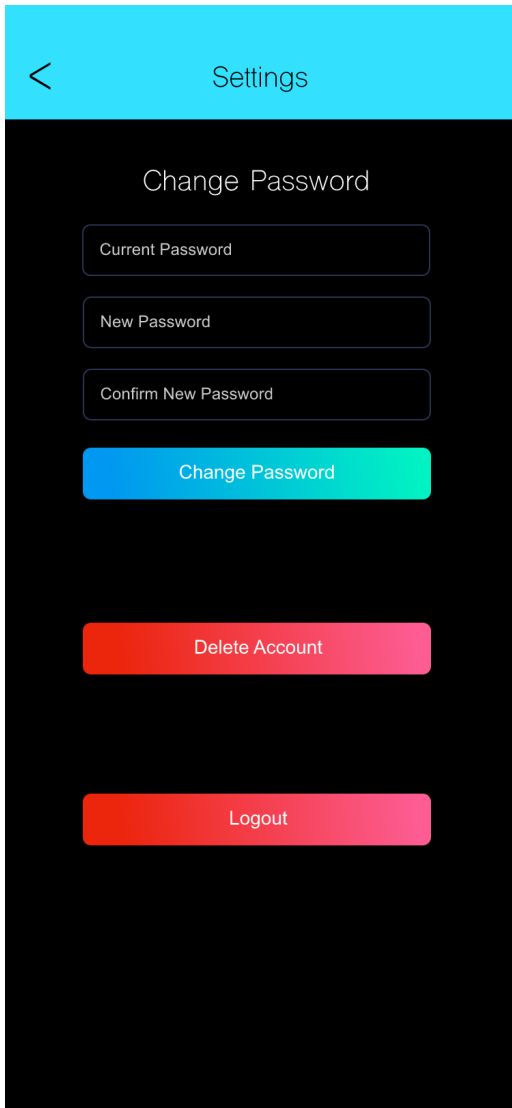


Figure 12: Settings Interface.

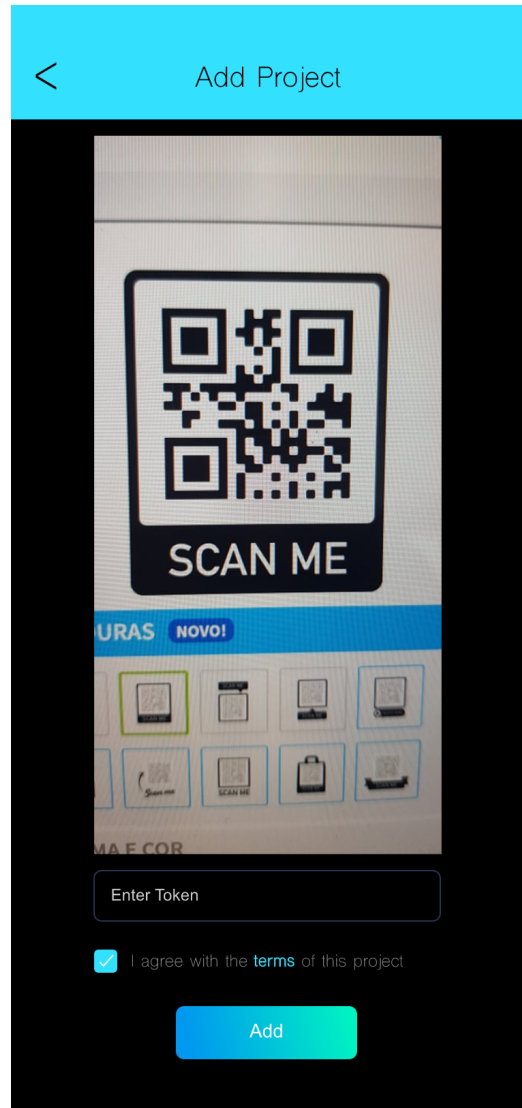


Figure 13: Add Project User Interface.

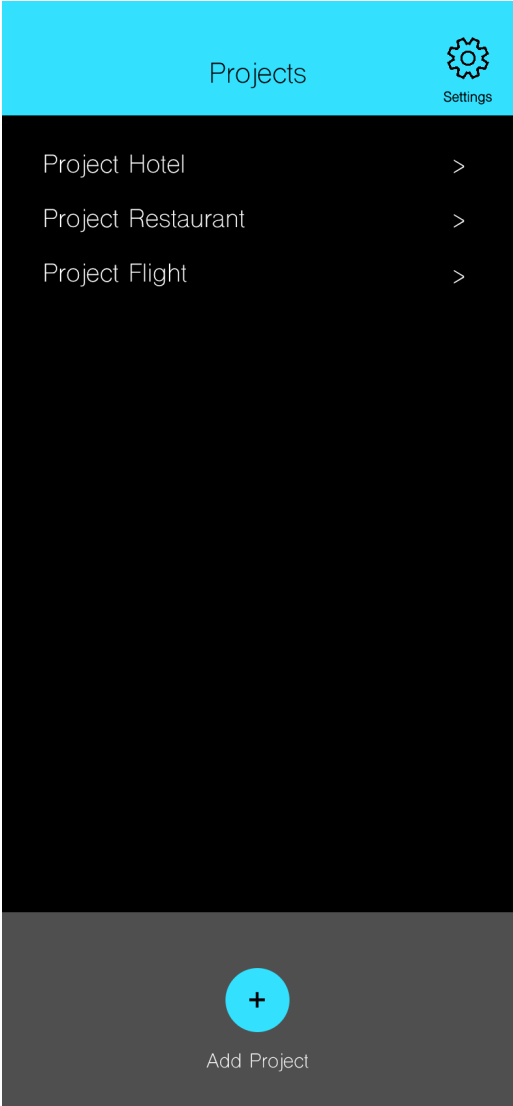


Figure 14: Choose Project User Interface.

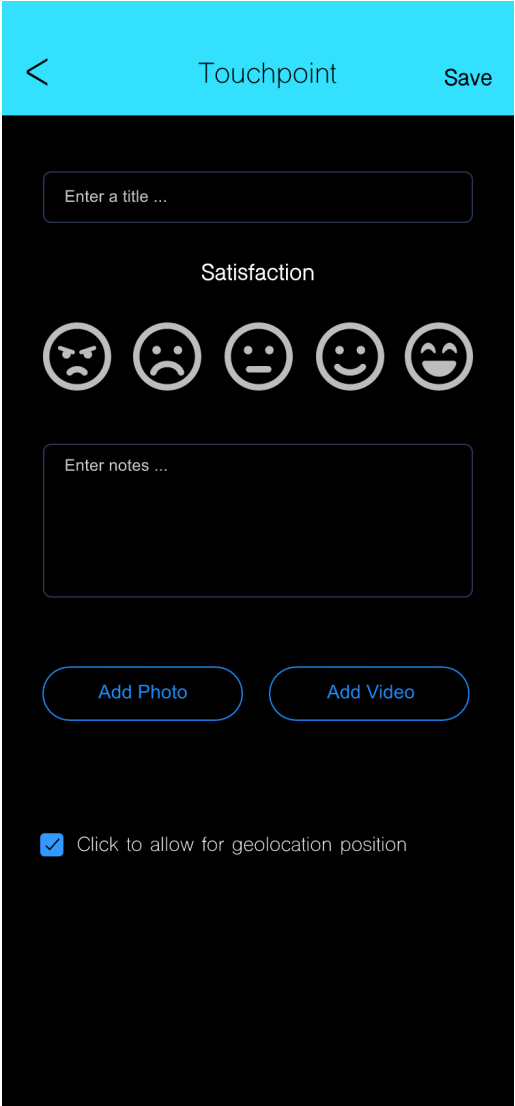


Figure 15: New/Edit Touchpoint User Interface.

This page is intentionally left blank.

Appendix C - Testing Results and Descriptions

ID	Test	Status	Tool
1	Change Password With Correct Credentials	Pass	Selenium IDE
2	Change Password With Wrong Password	Pass	Selenium IDE
3	Create a Project	Pass	Selenium IDE
4	Delete Account	Pass	Selenium IDE
5	Delete Account With Wrong Password	Pass	Selenium IDE
6	Login With Correct Credentials	Pass	Selenium IDE
7	Login With Participant Account	Pass	Selenium IDE
8	Login With Wrong Email	Pass	Selenium IDE
9	Login With Wrong Password	Pass	Selenium IDE
10	Register	Pass	Selenium IDE
11	Register With Participant Account	Pass	Selenium IDE
12	Drag Touchpoint to Correct Place	Pass	Done By The Author
13	Drag Touchpoint to Wrong Place	Pass	Done By The Author
14	Filter By All Satisfactions In Map	Pass	Done By The Author
15	Filter By User In Map	Pass	Done By The Author
16	Add New Tag To A Project	Pass	Done By The Author
17	Add Tag To A Touchpoint	Pass	Done By The Author
18	Delete Tag From Project	Pass	Done By The Author

Table 26: End-to-end testing conclusions

ID	Description
1	Click on Button To Change Password Click on Current Password field Type Current Password Click on New Password field Type New Password Click on New Password Confirmation field Type New Password Click on Change Password Button Assert Alert "Password Changed Successfully"

Table 27: Description of Test with ID 1

ID	Description
2	Click on Button To Change Password Click on Current Password field Type Wrong Current Password Click on New Password field Type New Password Click on New Password Confirmation field Type New Password Click on Change Password Button Assert Alert "The current password is wrong"

Table 28: Description of Test with ID 2

ID	Description
3	Click on Name of Project field Type the name of the project Click on Create Project Button Assert window.location.href=url/project

Table 29: Description of Test with ID 3

ID	Description
4	Click on Delete Account Section Click on Delete Account Button Click on Enter Password field Type Current Password Click on Delete Account Button Assert window.location.href=url/

Table 30: Description of Test with ID 4

ID	Description
5	Click on Delete Account Section Click on Delete Account Button Click on Enter Password field Type Wrong Current Password Click on Delete Account Button Assert alert "The password is wrong"

Table 31: Description of Test with ID 5

ID	Description
6	Click on email field Type current email Click on password field Type current password Click on sign in button Assert window.location.href=url/home

Table 32: Description of Test with ID 6

ID	Description
7	Click on email field Type email of a participant account Click on password field Type password Click on sign in button Assert alert "This is a participant account"

Table 33: Description of Test with ID 7

ID	Description
8	Click on email field Type email of a non existent account Click on password field Type password Click on sign in button Assert alert "User not found"

Table 34: Description of Test with ID 8

ID	Description
9	Click on email field Type current email Click on password field Type wrong password Click on sign in button Assert alert "Wrong password"

Table 35: Description of Test with ID 9

ID	Description
10	Click on sign up button Click on email field Type email Click on password field Type password Click on sign up button Assert window.location.href=url/home

Table 36: Description of Test with ID 10

ID	Description
11	Click on sign up button Click on email field Type email for a participant account Click on password field Type password Click on sign up button Assert alert "Email already in use"

Table 37: Description of Test with ID 11

ID	Description
12	Click and Drag touchpoint Drop it in the same row Reload the page Visualize if it is in the same place

Table 38: Description of Test with ID 12

ID	Description
13	Click and Drag touchpoint Drop it in the different row Visualize if it is in the initial dragging place Reload the page Visualize if it is in the initial place

Table 39: Description of Test with ID 13

ID	Description
14	Click on a satisfaction checkbox Visualize in the map if all touchpoints correspond to the correct spot

Table 40: Description of Test with ID 14

ID	Description
15	Click on a satisfaction checkbox Select a user in the dropdown box Visualize if all touchpoints of that user correspond to the correct spot

Table 41: Description of Test with ID 15

ID	Description
16	Click on Add Tags to Project Add tag with the name Test Tag Click on Tags Visualize if the tag Test Tag is show

Table 42: Description of Test with ID 16

ID	Description
17	Double Click in a Touchpoint Click on the dropdown with the name tags Select teste tag Reload the page Visualize if tag is shown in that touchpoint

Table 43: Description of Test with ID 17

ID	Description
18	Click on Tags Click on X button of a Tag Click the button Yes Click on Journey Maps Button Double Click on a Touchpoint that had the deleted tag Visualize if tag is not shown

Table 44: Description of Test with ID 18

This page is intentionally left blank.

Software Requirements Specification
for
A software tool for customer experience evaluation

Prepared by Diogo Boinas

University of Coimbra
Dissertation in the context of the Master in Informatics Engineering

Specialization in Software Engineering advised by Prof. Paulo Rupino da Cunha

Coimbra, 5. September 2022

Contents

Revision History	1
1 Introduction	2
1.1 Purpose	2
1.2 Intended Audience	2
1.3 Intended Use	2
1.4 Product Scope	2
1.5 Risk Definition	3
2 Overall Description	4
2.1 Product Perspective	4
2.1.1 System Interfaces	4
2.1.2 Interfaces	4
2.1.3 Hardware Interfaces	5
2.1.4 Software Interfaces	5
2.1.5 Memory Constraints	5
2.2 Product Functions	6
2.3 User Characteristics	7
2.4 Constraints	7
3 Specific Requirements	8
3.1 Use Cases Model	8
3.1.1 Context Diagram	8
3.1.2 Use Cases Diagram	9
3.1.3 Use Cases	12
3.2 User Interface Model	25
3.2.1 User Experience Diagram	26
3.2.2 User Interface	28
3.3 Non Functional Requirements	37
3.4 Requirements List	38

3.4.1	Participant Requirements	38
3.4.2	Researcher Requirements	38

Revision History

Revision	Date	Author(s)	Description
1.0	19.11.2021	B. Diogo	Chapter 1 - Introduction, Chapter 2 - Overall Description
1.1	20.12.2021	B. Diogo	Chapter 3- Specific Requirements
1.2	29.12.2021	B. Diogo	Full Document Revision

Chapter 1

Introduction

In this chapter, we are going to provide an overview of the requirements document.

1.1 Purpose

The purpose of this document is to build mobile application and a web application, and this document will present the requirements for both.

1.2 Intended Audience

At first the intended audience of this software will be the students of the Department of Informatics Engineering at University of Coimbra, more concretely the students taking the course of Service Engineering that will be the first ones to test the platform. This software will be build for the Master in Informatics Engineering dissertation and advised by Prof. Paulo Rupino da Cunha.

1.3 Intended Use

The intended use of this platform will be for the students of Service Engineering learn with a hands-on approach about the themes: Mobile Ethnography and Service Design.

1.4 Product Scope

The product scope of this platform will be to do Mobile Ethnography. It will consist in a website for the researchers to create projects, collect and gather data and finally analyse it. It will also have a mobile application that will be used by the participants and then they can send their behaviour, opinions, etc through the application for the researcher to later analyse it. Benefits of this software is that ethnographers can have an all in one system, not having to use several softwares to do their studies. Another benefit is the reduced cost, and also the time spent, comparing this platform to one-on-one ethnography. Here, the researcher basically just has to create a project and publicize it to have success. On a one-to-one research, the researcher has to go along with the participant all the time. This is time consuming and costly.

1.5 Risk Definition

There is no risk associated with this product, however if we can't make all the requirements stated in this document it can become a risk of the thesis that is being developed can't reach success in the implementation part. However, it could be continued by another student next year.

Chapter 2

Overall Description

In this chapter, we are going to describe the general factors that affect the product and its requirements.

2.1 Product Perspective

2.1.1 System Interfaces

Starting with the backend, there will be a cloud-based database, Cloud Firestore, that stores the following information:

1. **User Data** - It serves to store information about the user, if they are participants or researchers, their name and age.
2. **Collected Data** - It serves to store information about the collected data about each user for each project.

There will also be a storage part in the cloud that will serve to store pictures and videos.

Moving on now to the frontend, there will be a progressive web application to collect data, that will be built for the purpose of working in Android and IOS using a cross-platform framework, React Native. There will also be a website to analyse data, that will be built using React.js.

2.1.2 Interfaces

In this section, we will show how the system works. Having the website (web application), the progressive web application (mobile app) and the backend (firebase). The following figure, 2.1, will show the system in a brief perspective.

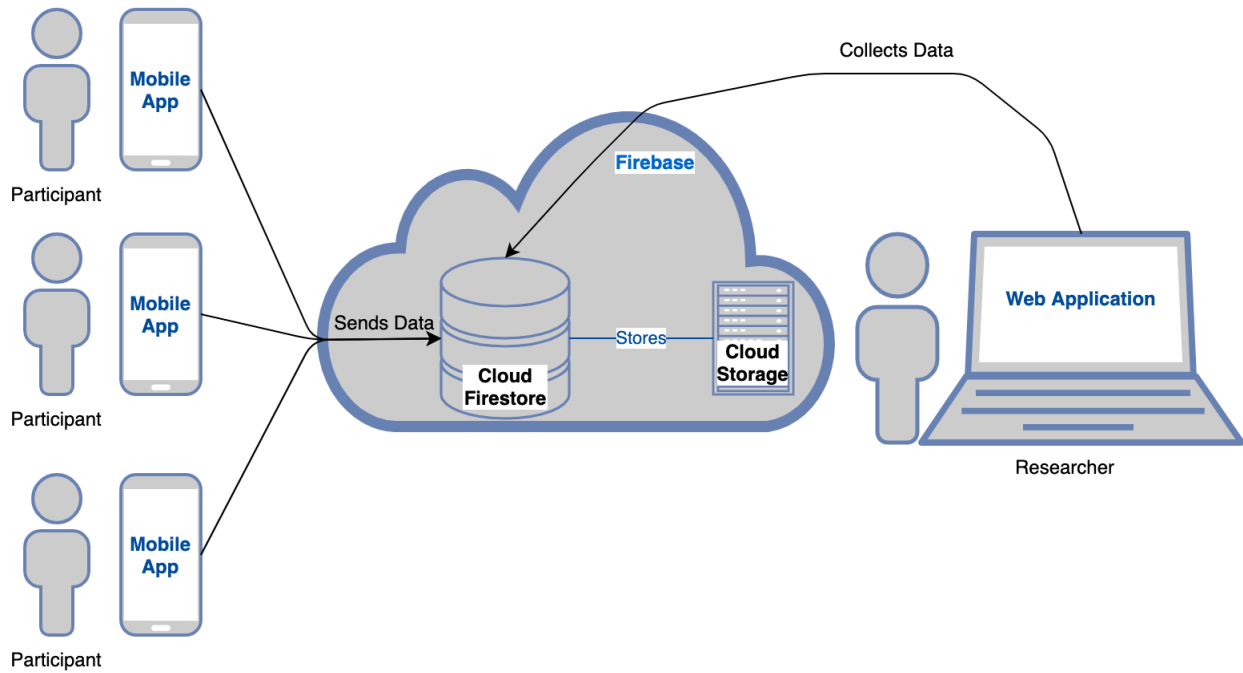


Figure 2.1: Interface of the system

In this figure, we can see the several participants in the left side, there are three in this case, but there can be much more. They are sending the data to the cloud firestore, that is a NoSQL database, that is in the cloud. The cloud in question is Firebase from Google. We can also see the storage part, that is also in the cloud. This storage will contain all the pictures and videos from the participants. Finally, we can see in the right side of the figure the researchers, he collects data to later analyse it.

The mobile application and the web application are Graphical User Interfaces. Firebase is as PaaS (Platform as a Service) and includes cloud firestore that is a database and cloud storage that is a storage in the cloud.

2.1.3 Hardware Interfaces

The mobile application has to work in all recent iPhones and in the most recent android phones. So there will need to be attention to detail for example in cases where the screen has a different shape, like the iPhone X. To sum up, it has to be responsive for almost all mobile phones.

The web application also has to be responsive, to work seamlessly in all computer devices. This is 16:9 and 16:10 in the majority of computer monitors.

2.1.4 Software Interfaces

As a design choice, we will be using Google's Firebase and to connect with both of our applications we will use their SDK (Software Development Kit).

2.1.5 Memory Constraints

There are no memory constraints in this case, due to being a pay-as-you-go platform. We just have to pay attention to the number of reads and writes, and optimize it. If we need more memory or speed, we can always have it.

2.2 Product Functions

This software will have two distinct areas:

- Website Application
- Mobile Application

In this section we are going to do a summary of the major functions that the software will perform in a high level.

Website Application

The website application will be for researchers, this means, the person that is collecting the data for later analysing it. The researcher can sign up on the website and then, to access the functionalities of the software tool, it has to sign in.

After signing in, the researcher can create a new project, and then he receives a QR code to send to the participants and invite them to join the specific project. Also, the researcher can view his projects, can finalize and archive them. Another functionality is that when the researcher clicks on the project, he enters the project page.

After entering the project page, the researcher can view the data collected filtered by the different participants. The project page is divided in three different functions, data that is the first page shown when the researcher enters the project, map that shows a map with the location of the different touchpoints collected and the overview where the researcher can manipulate the data. After the data is manipulated, the data page shows the data already manipulated.

The researcher can also export the data in the data page to CSV format.

Mobile Application

The application will be for participants who want to join any project. They can sign up and sign in.

After signing in, the participant can scan a QR code and enter a project. Inside the project, the participant can submit touchpoints. The touchpoint includes:

- **Title** - It is a name for the touchpoint, in this case just a brief name.
- **Emotion** - Rating from 1 to 5 with smiley faces, 1 being the worse and 5 being the best.
- **Notes** - In this field, the participant can detail what exactly happened.
- **Media** - Can add photos or videos of the situation.
- **Location** - The participant can choose to send his GPS coordinates or not.

2.3 User Characteristics

The users who will use this software tool are two types of users. One type of user will be the general public. These users are the ones who will use the mobile application. They can be people who do not have any experience with new technologies, so the front-end has to be intuitive and straightforward. The other type of user will be someone who works for a company and is in charge of analysing the data collected. A user guide will be provided on the website so that these types of users can learn all the functionalities.

2.4 Constraints

In order to comply to regulatory policies, this software tool has to comply to GDPR. We have to make sure this is achieved before putting the software in a production environment. If this isn't accomplished, large fines can be applied.

There are other limitations, such as hardware. We are going to host the website in DEI virtual private servers. These servers have a limit on the amount of storage, the number of cores of the processor, and the ram memory. If we have too many users, this is a constraint and can make the website slow to load or even crash.

By using cloud firestore database, we also have to write security rules to prevent attacks to the database.

Chapter 3

Specific Requirements

3.1 Use Cases Model

In this chapter, the actors, use cases and their relationships will be presented. We want to know how different users behave with regard to the system, with the purpose of solving any problem. The actors will have goals that can not always be fulfilled. We also have the interactions between the user and the system. Having this, we could know how to reach the goals. In the use cases diagram, we can visually see what the goals are. Simplifying the final solution, which are the use cases themselves.

3.1.1 Context Diagram

This diagram presents the interaction between the primary actors on the left side and the system. There are no external actors; however, if they existed, they would be presented on the right side of the system.

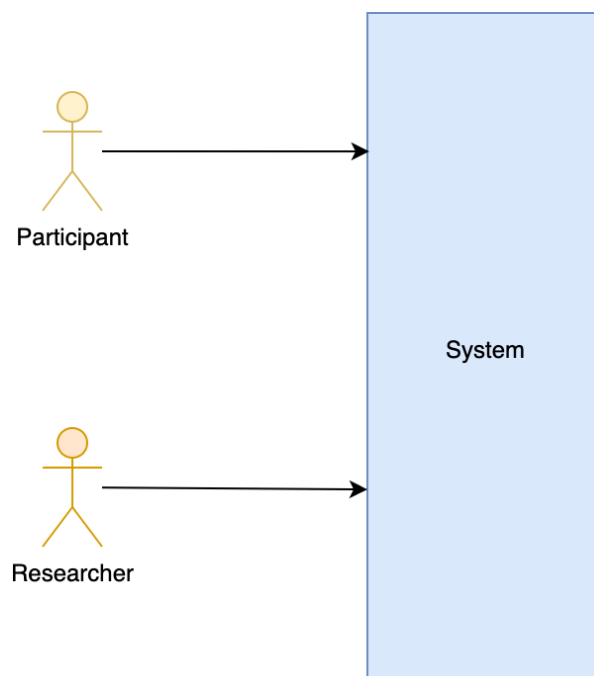


Figure 3.1: Context Diagram.

3.1.2 Use Cases Diagram

The following diagrams will show several functionalities for the actors of our applications.

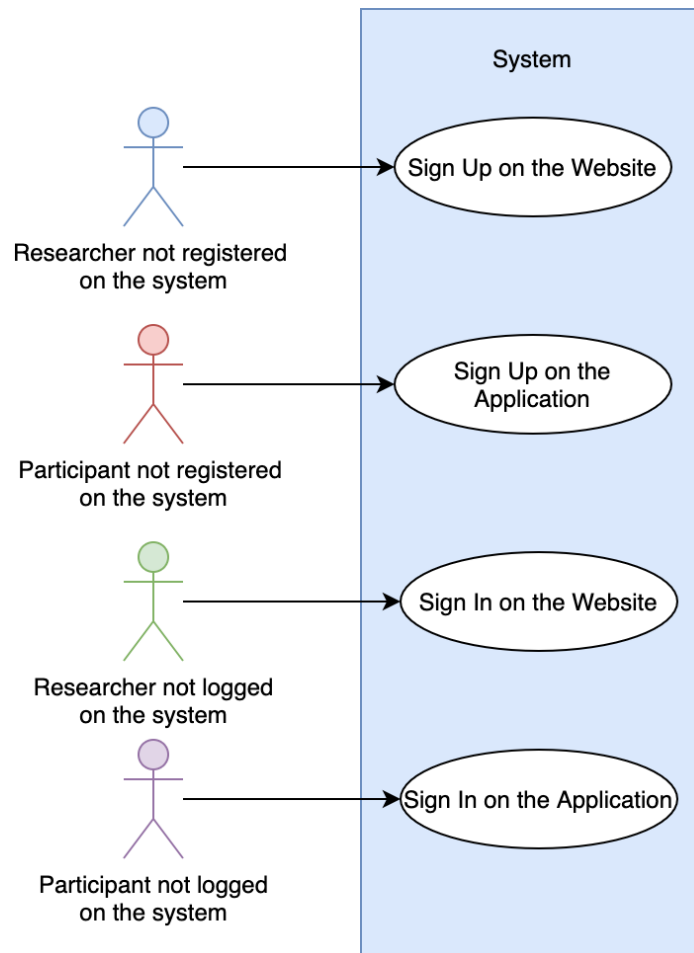


Figure 3.2: Use Cases Diagram for Signing In and Signing Up.

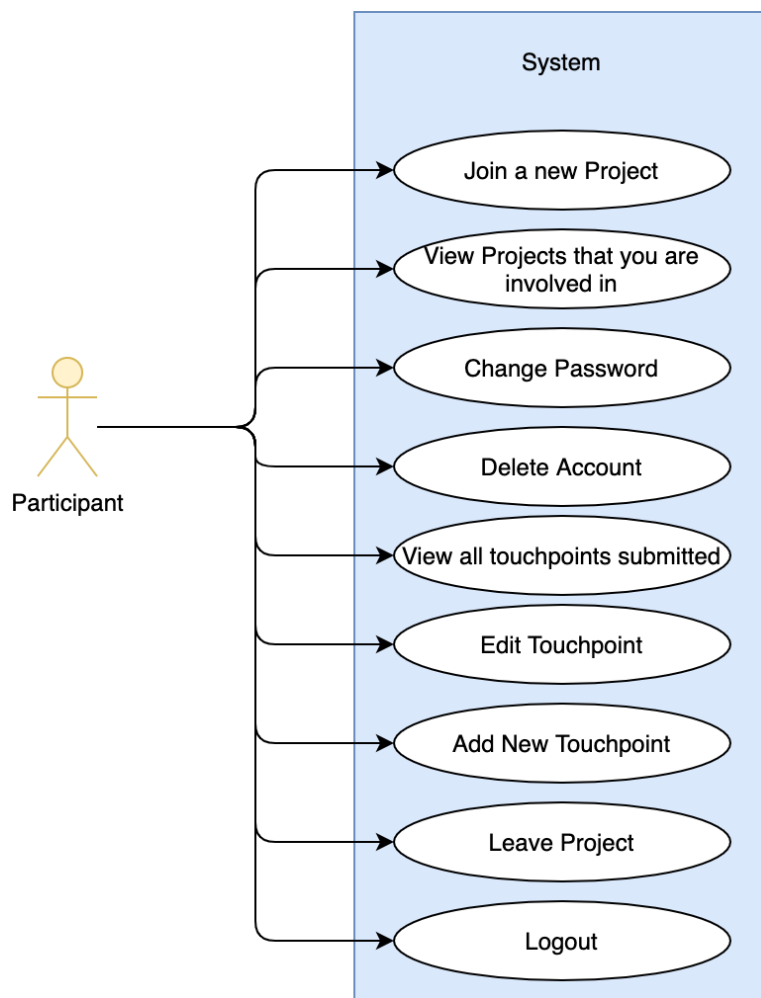


Figure 3.3: Use Cases Diagram for Participant Functionalities.

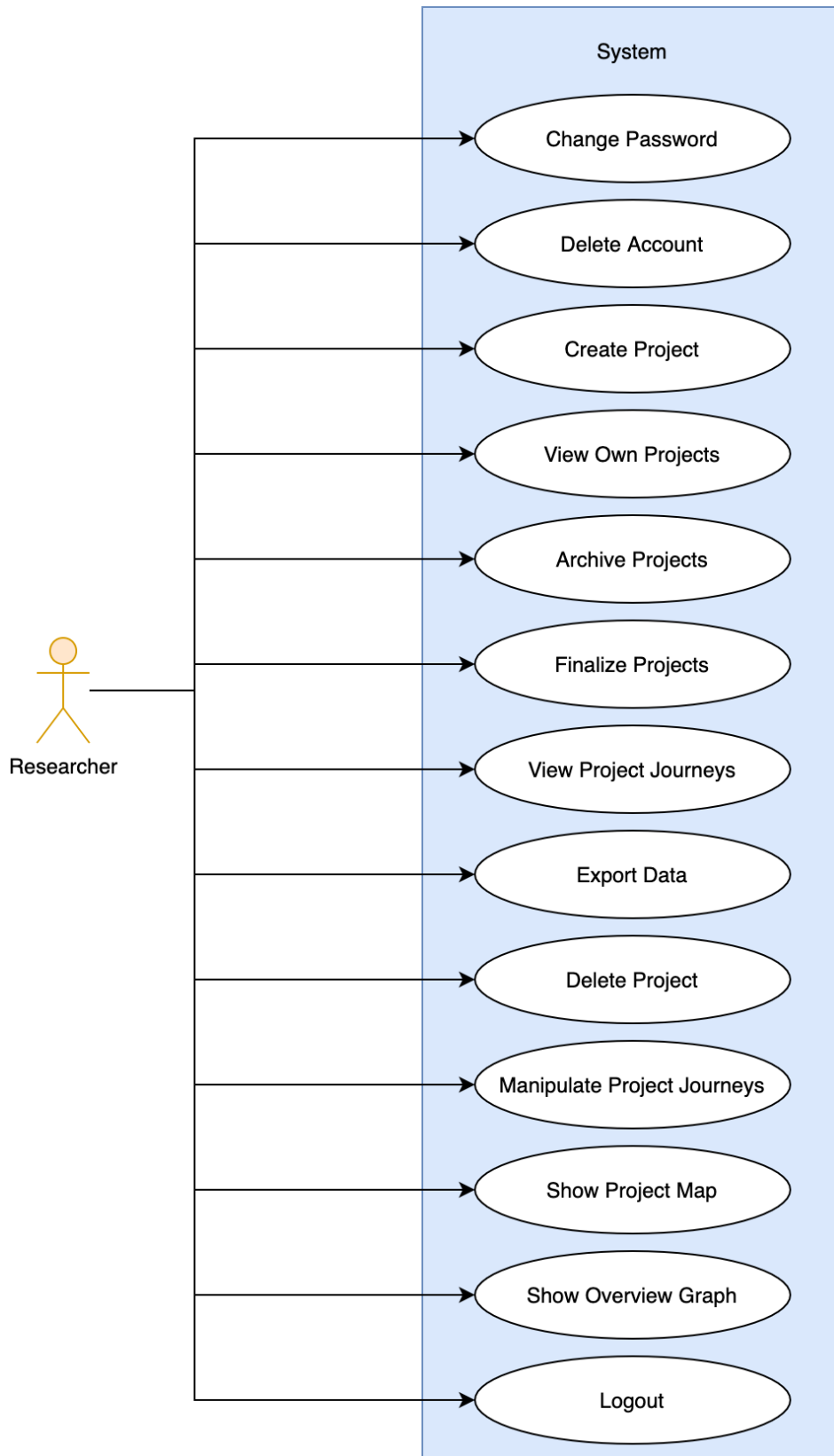


Figure 3.4: Use Cases Diagram for Researcher Functionalities.

3.1.3 Use Cases

In this section of the use cases, we will describe the behaviour of the primary actors regarding the functionalities presented in the various use cases diagrams.

Participant Sign Up

A new participant saw the application on the internet and he wants to sign up on it.

Name	Participant Sign Up
ID	01
Primary Actor	Participant not registered on the System
Level	Blue
Description	The participant makes an account on the system, inserting his personal data
Pre-condition	The participant is not registered and has the application installed on his device
Basic path	<ol style="list-style-type: none"> 1: The participant opens the application 2: The participant selects the Sign Up option 3: The participant Inserts the personal data (email and password) 4: The participant accepts the legal terms and conditions 5: The Participant selects the Sign Up option 6: The System verifies the data 7: The System validates the data 8: The System adds the new participant
Alternative path	<ol style="list-style-type: none"> 6.a: The System finds an error in the participant data 6.b: The System informs the participant about the error (email already exists or password too weak) 7.a: The system notifies to change the personal data
Post-condition	Participant is registered and he is redirected to the Choose Project page
Frequency	Medium

Table 3.1: Participant Sign Up Use Case Table

Researcher Sign Up

A researcher was told about the application by a friend and he wants to sign up on it.

Name	Researcher Sign Up
ID	02
Primary Actor	Researcher not registered on the System
Level	Blue
Description	The researcher makes an account on the system, inserting his personal data
Pre-condition	The researcher is not registered
Basic path	<ol style="list-style-type: none"> 1: The researcher goes to the system's website 2: The researcher selects the Sign Up option 3: The researcher inserts the personal data (email and password) 4: The participant accepts the legal terms and conditions 5: The Participant selects the Sign Up option 6: The System verifies the data 7: The System validates the data 8: The System adds the new researcher
Alternative path	<ol style="list-style-type: none"> 6.a: The System finds an error in the researcher data 6.b: The System informs the researcher about the error (email already exists or password too weak) 7.a: The system notifies to change the personal data
Post-condition	Researcher is registered and he is redirected to the Dashboard page
Frequency	Medium

Table 3.2: Researcher Sign Up Use Case Table

Participant Sign In

A participant wants to sign in.

Name	Participant Sign In
ID	03
Primary Actor	Participant
Level	Blue
Description	The participant signs in inserting his personal data
Pre-condition	The participant is already registered on the system
Basic path	1: The participant opens the application 2: The participant selects the Sign In option 3: The participant inserts the personal data (email and password) 4: The participant selects the Sign In option 5: The System verifies the data
Alternative path	5.a: The System finds an error in the user data (email doesn't exist or wrong password) 5.b: The System informs the user about the error
Post-condition	Participant is signed in and he is redirected to the Choose Project page
Frequency	Medium

Table 3.3: Participant Sign In Use Case Table

Researcher Sign In

A researcher wants to sign in.

Name	Researcher Sign In
ID	04
Primary Actor	Researcher
Level	Blue
Description	The researcher signs in inserting his personal data
Pre-condition	The researcher is already registered on the system
Basic path	1: The researcher goes to the system's website 3: The researcher inserts the personal data (email and password) 4: The researcher selects the Sign In option 5: The System verifies the data
Alternative path	5.a: The System finds an error in the user data (email doesn't exist or wrong password) 5.b: The System informs the user about the error
Post-condition	Researcher is logged in and he is redirected to the Choose Project page
Frequency	Medium

Table 3.4: Researcher Sign In Use Case Table

Join a New Project

A participant enters a flight and sees the QR code to enter the airline's project. Now, he wants to join this project to give his insights to the airline.

Name	Join a New Project
ID	05
Primary Actor	Participant
Level	Blue
Description	The participant joins a new project
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the Add Project option 3: The participant scans the QR Code 4: The participant selects the Add option 5: The System inserts the participant in that project
Alternative path	3.a: The participant enters the token for the project
Post-condition	Participant is redirected to the Choose Projects page
Frequency	Medium

Table 3.5: Join a New Project Use Case Table

View Projects that you are Involved In

A participant wants to see the projects that he has entered.

Name	View Projects that you are Involved In
ID	06
Primary Actor	Participant
Level	Blue
Description	The participant wants to see the projects that he has entered
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant sees all the projects he has previous entered
Post-condition	Participant sees all projects that he is involved in
Frequency	High

Table 3.6: Views Project that you are Involved In Use Case Table

Change Password

A participant wants to change his password.

Name	Change Password
ID	07
Primary Actor	Participant
Level	Blue
Description	The participant wants to change his password
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant inserts his current password 4: The participant inserts his new password 5: The participant inserts his new password for confirmation 6: The participant selects the change password option 7: The system changes the participant's password
Alternative path	7.a: The system informs the participant to choose another password (inserted password is too weak)
Post-condition	Participant's password is changed
Frequency	Low

Table 3.7: Change Password Use Case Table

Delete Account

A participant wants to delete his account.

Name	Delete Account
ID	08
Primary Actor	Participant
Level	Blue
Description	The participant wants to delete his account
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant selects the delete account option 4: The participant inserts his password 5: The system informs the participant about the successful deletion
Alternative path	5.a: The system informs the participant that his password is wrong
Post-condition	Participant's account is deleted
Frequency	Low

Table 3.8: Delete Account Use Case Table

View all touchpoint points submitted

A participant wants to see all the touchpoints submitted for a particular project.

Name	View All Touchpoints Submitted
ID	09
Primary Actor	Participant
Level	Blue
Description	The participant wants to see all the touchpoints submitted for a project
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project we wants to see the touchpoints 3: The participant views all touchpoints submitted
Alternative path	None
Post-condition	Participant is on the project page and views all the touchpoints he submitted for a particular project
Frequency	High

Table 3.9: View all touchpoint points submitted Use Case Table

Edit touchpoint

A participant wants to change the photo for a specific touchpoint.

Name	Edit Touchpoint
ID	10
Primary Actor	Participant
Level	Blue
Description	The participant wants to change the photo for a touchpoint
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the project we wants to change the touchpoint 3: The participant selects the specific touchpoint 4: The participant selects the add photo option 5: The participant selects the new photo in his device 6: The participant selects the save option
Alternative path	None
Post-condition	Touchpoint is modified with success and the participant is redirected to the choose touchpoints pag
Frequency	Low

Table 3.10: Edit Touchpoint Use Case Table

Add New Touchpoint

A participant wants to add a new touchpoint.

Name	Add New Touchpoint
ID	11
Primary Actor	Participant
Level	Blue
Description	The participant wants to add a new touchpoint
Pre-condition	The participant is already registered and signed in on the system
Basic path	<ol style="list-style-type: none"> 1: The participant opens the application 2: The participant selects the project we wants to add the touchpoint 3: The participant selects the new touchpoint option 4: The participant enters the title 5: The participant selects the satisfaction 6: The participant writes a small description in the notes section 7: The participant selects the save option
Alternative path	<ol style="list-style-type: none"> 6.a: The participant selects the Add Photo Option 6.b: The participant selects the photo in his device 6.1.a: The participant selects the Add Video Option 6.1.b: The participant selects the video in his device 6.2: The participant selects the option for allowing the geolocation position
Post-condition	Touchpoint is submitted and the participant is redirected to the choose touchpoints page
Frequency	Normal

Table 3.11: Edit Touchpoint Use Case Table

Leave Project

A participant wants to leave a project.

Name	Leave Project
ID	12
Primary Actor	Participant
Level	Blue
Description	The participant wants to leave a project
Pre-condition	The participant is already registered and signed in on the system
Basic path	<ol style="list-style-type: none"> 1: The participant opens the application 2: The participant selects the project we wants to leave 3: The participant selects the leave option
Alternative path	None
Post-condition	Participant is removed from the project and is redirected to the choose project page
Frequency	Low

Table 3.12: Leave Project Use Case Table

Participant's Logout

A participant wants to logout from the application.

Name	Participant's Logout
ID	13
Primary Actor	Participant
Level	Blue
Description	The participant wants to logout from the application
Pre-condition	The participant is already registered and signed in on the system
Basic path	1: The participant opens the application 2: The participant selects the settings option 3: The participant selects the logout option
Alternative path	None
Post-condition	Participant is signed out of the application and is redirected to the homescreen page
Frequency	Low

Table 3.13: Participant's Logout Use Case Table

Change Password

A researcher wants to change his password.

Name	Change Password
ID	14
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to change his password
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher inserts his current password 3: The researcher inserts his current password 4: The researcher inserts his new password 5: The researcher inserts his new password for confirmation 6: The researcher selects the change password option 7: The system changes the researcher's password
Alternative path	7.a: The system informs the participant to choose another password (inserted password is too weak)
Post-condition	Researcher's password is changed
Frequency	Low

Table 3.14: Change Password Use Case Table

Delete Account

A researcher wants to delete his account.

Name	Delete Account
ID	15
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to delete his account
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the delete account option 3: The researcher inserts his password 4: The system informs the researcher about the successful deletion
Alternative path	7.a: The system informs the researcher that his password is wrong
Post-condition	Researcher's account is deleted
Frequency	Low

Table 3.15: Delete Account Use Case Table

Create Project

A researcher wants to create a new project.

Name	Create Project
ID	16
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to create a new project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the create project option 3: The researcher inserts the name of the project and the legal terms 4: The researcher selects the create project option 5: The system creates the new project
Alternative path	None
Post-condition	Researcher is redirected to project created page and the project token and QR Code is displayed
Frequency	Normal

Table 3.16: Create Project Use Case Table

View Own Projects

A researcher wants to view his own projects.

Name	View Own Projects
ID	17
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to view his own projects
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option
Alternative path	None
Post-condition	Researcher is redirected to view projects page and all the projects are displayed
Frequency	High

Table 3.17: View Own Projects Use Case Table

Finalize Projects

A researcher wants to finalise a project to not collect more data.

Name	Finalize Projects
ID	18
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to finalize a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the finalize option of the specific project
Alternative path	None
Post-condition	Project is moved from the active to the finalized section
Frequency	Normal

Table 3.18: Finalize Projects Use Case Table

Archive Projects

A researcher wants to archive a project because all of the data is analyzed.

Name	Archive Projects
ID	19
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to archive a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the archive option of the specific project
Alternative path	None
Post-condition	Project is moved from the finalized to the archived section
Frequency	Normal

Table 3.19: Finalize Projects Use Case Table

View Project Journeys

A researcher wants to analyze the project journey of a specific project.

Name	View Project Journeys
ID	20
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to analyze a project journeys
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page
Alternative path	None
Post-condition	Project journeys are displayed
Frequency	High

Table 3.20: View Project Journeys Use Case Table

Export Data

A researcher wants to export data from a specific project.

Name	Export Data
ID	21
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to export data from a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the export data option
Alternative path	None
Post-condition	Data is downloaded in CSV format
Frequency	Low

Table 3.21: Export Data Use Case Table

Manipulate Project Journeys

A researcher wants to re-organize the order of the project journeys of the several participants for comparison.

Name	Manipulate Project Journeys
ID	22
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to to re-organize the order of the project journeys
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the data option 6: The researcher selects the Add option to enter an empty space between touchpoints 7: The researcher drags and drops a touchpoint to certain location 8: The researcher selects the save changes option
Alternative path	None
Post-condition	The new organization of data is saved in the database
Frequency	High

Table 3.22: Manipulate Project Journeys Use Case Table

Delete Project

A researcher wants to delete a project.

Name	Export Data
ID	23
Primary Actor	Researcher
Level	Blue
Description	A researcher wants to delete a project
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Delete Project option 6: The researcher inserts his password 7: The system deletes the project
Alternative path	7.a: The password inserted his wrong
Post-condition	Project is deleted from the database
Frequency	Low

Table 3.23: Delete Project Use Case Table

Show Project Map

A researcher wants to see the map showing all the touchpoints.

Name	Show Project Map
ID	24
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to see the map showing all the touchpoints
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Map option
Alternative path	None
Post-condition	The researcher is redirected to the Map Page
Frequency	High

Table 3.24: Show Project Map Use Case Table

Show Overview Graph

A researcher wants to see the overview graph.

Name	Show Overview Graph
ID	25
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to see the overview graph
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the view projects option 3: The researcher selects the specific project 4: The system redirects to the project page 5: The researcher selects the Overview option
Alternative path	None
Post-condition	The researcher is redirected to the Overview Page
Frequency	High

Table 3.25: Show Overview Graph Use Case Table

Researcher's Logout

A researcher wants to logout from the website.

Name	Researcher's Logout
ID	26
Primary Actor	Researcher
Level	Blue
Description	The researcher wants to logout from the website
Pre-condition	The researcher is already registered and signed in on the system
Basic path	1: The researcher opens the website 2: The researcher selects the logout option
Alternative path	None
Post-condition	Researcher is signed out of the website and is redirected to the homepage
Frequency	Normal

Table 3.26: Researcher's Logout Use Case Table

3.2 User Interface Model

To aid the development of the User Interface (UI), the first step was to build a User Experience Diagram (UED). This will help us to develop our interfaces into a coherent design. We can see in the UED the main focus areas, in other words, the several screens of our applications. We correlated the various focus areas to also understand navigation in our application. In each focus areas, we can see how the user interacts with the application, and also the types of data that are used. User Data is related to every type of data from the user of application, email, and password in our case. Project Data is all of the data related to project in a broader sense, name of the project, token, etc. Touchpoint Data is the specific data of each project, and it is more detailed than Project Data. It includes the several touchpoints of each project.

3.2.1 User Experience Diagram

Web Application UED

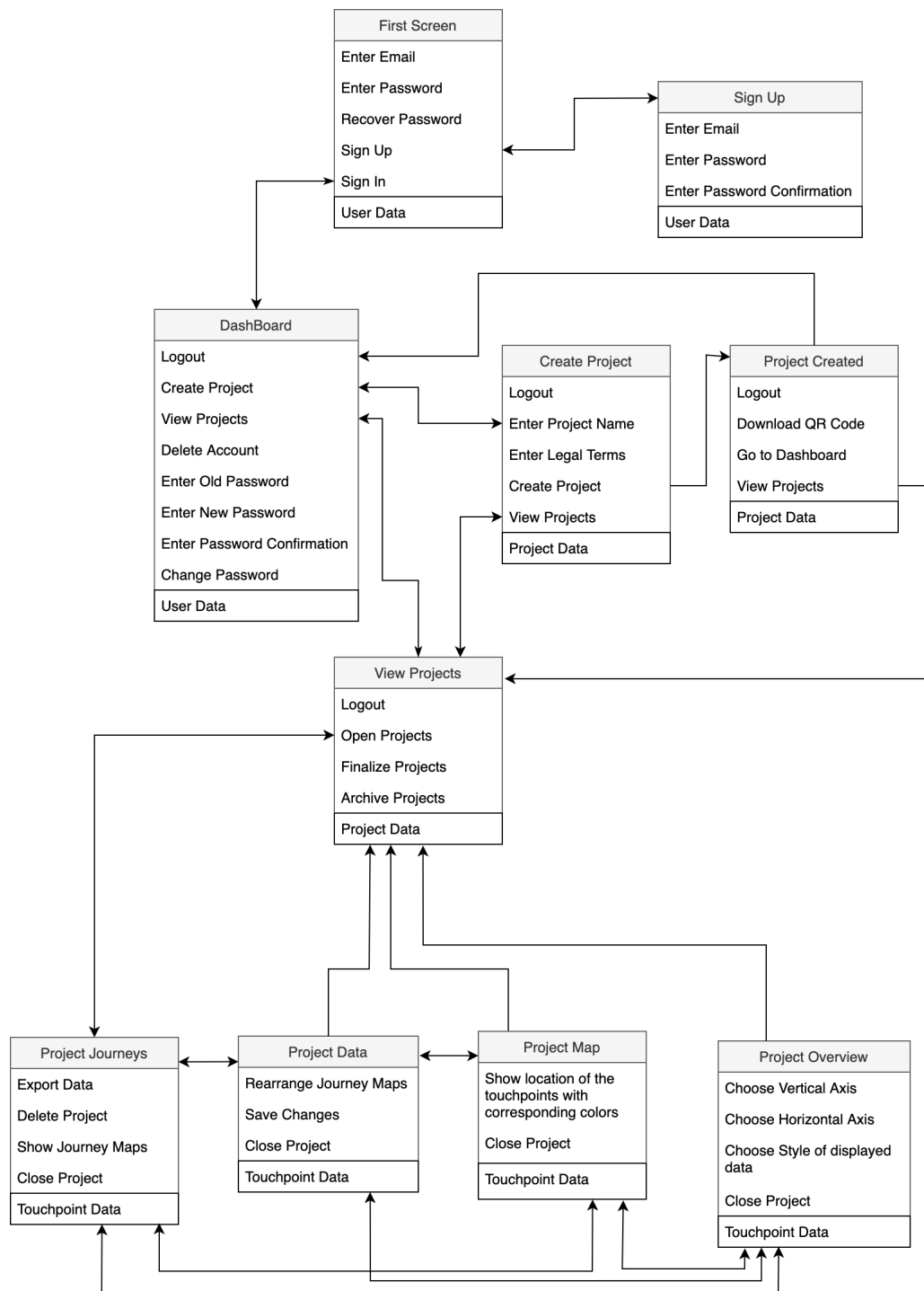


Figure 3.5: User Experience Diagram of Website

Mobile Application UED

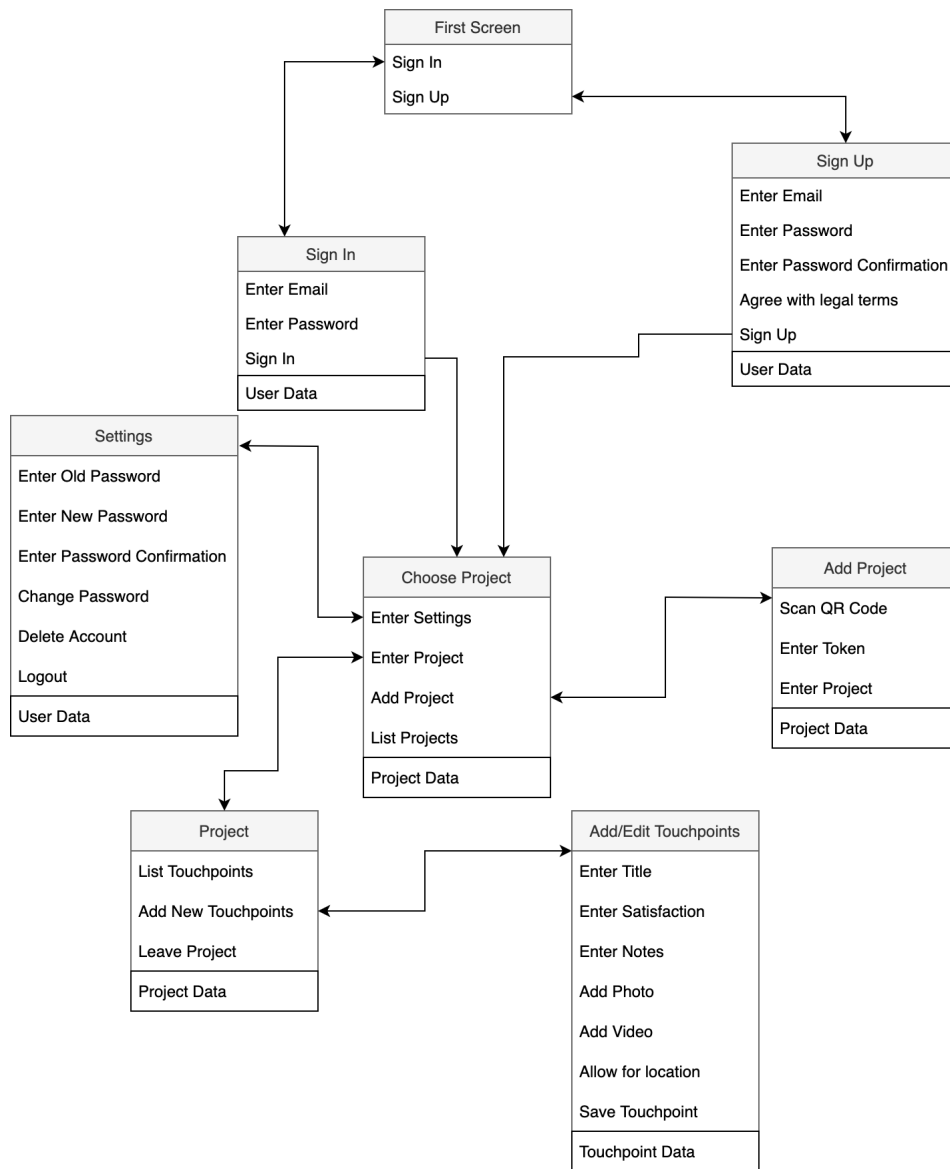


Figure 3.6: User Experience Diagram of Mobile Application

3.2.2 User Interface

Web Application

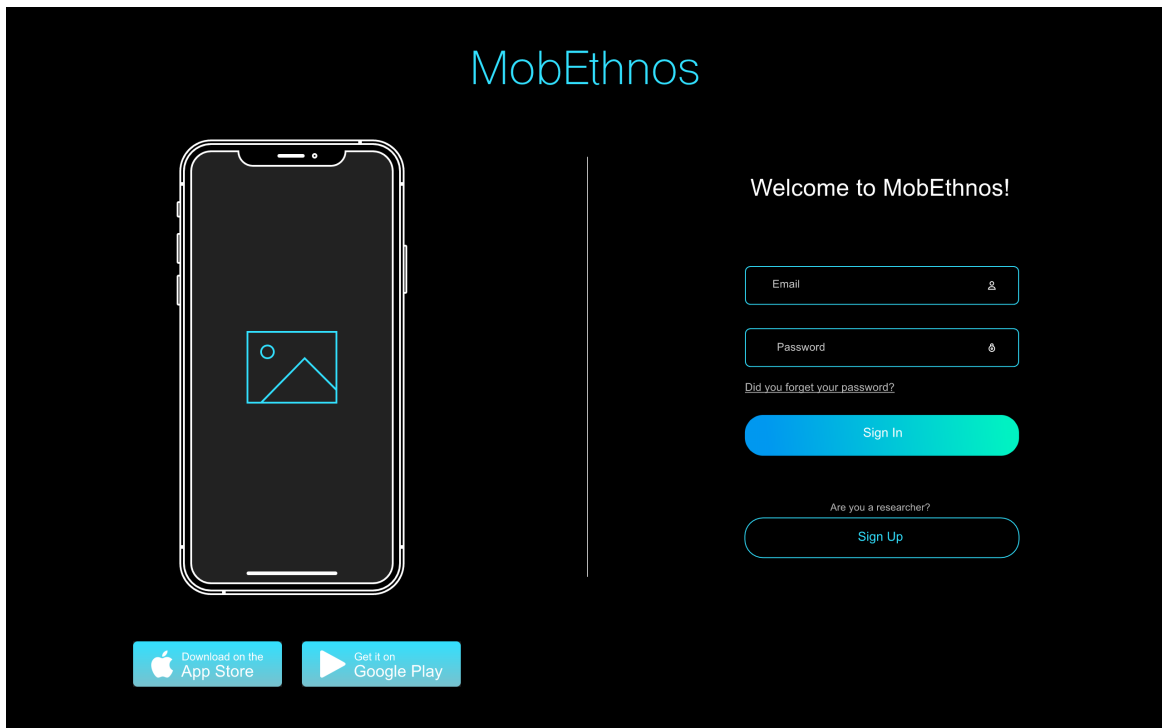


Figure 3.7: Homepage User Interface

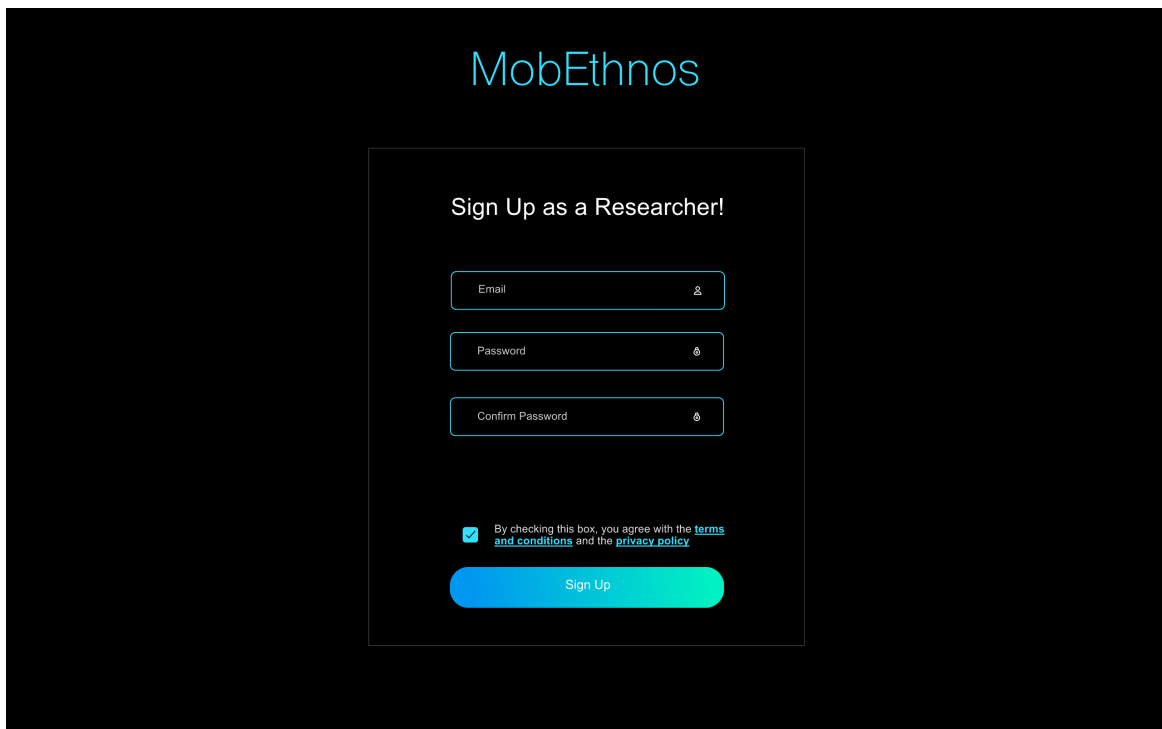


Figure 3.8: Sign Up User Interface

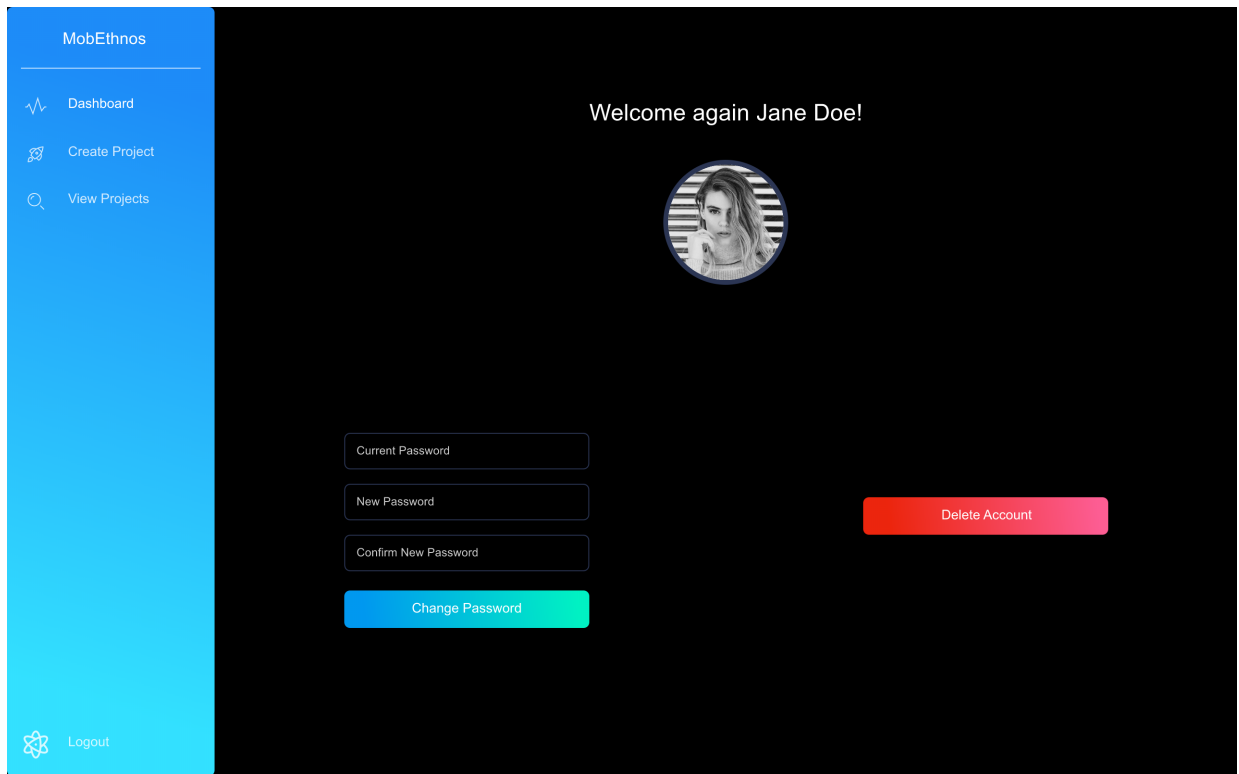


Figure 3.9: Dashboard User Interface

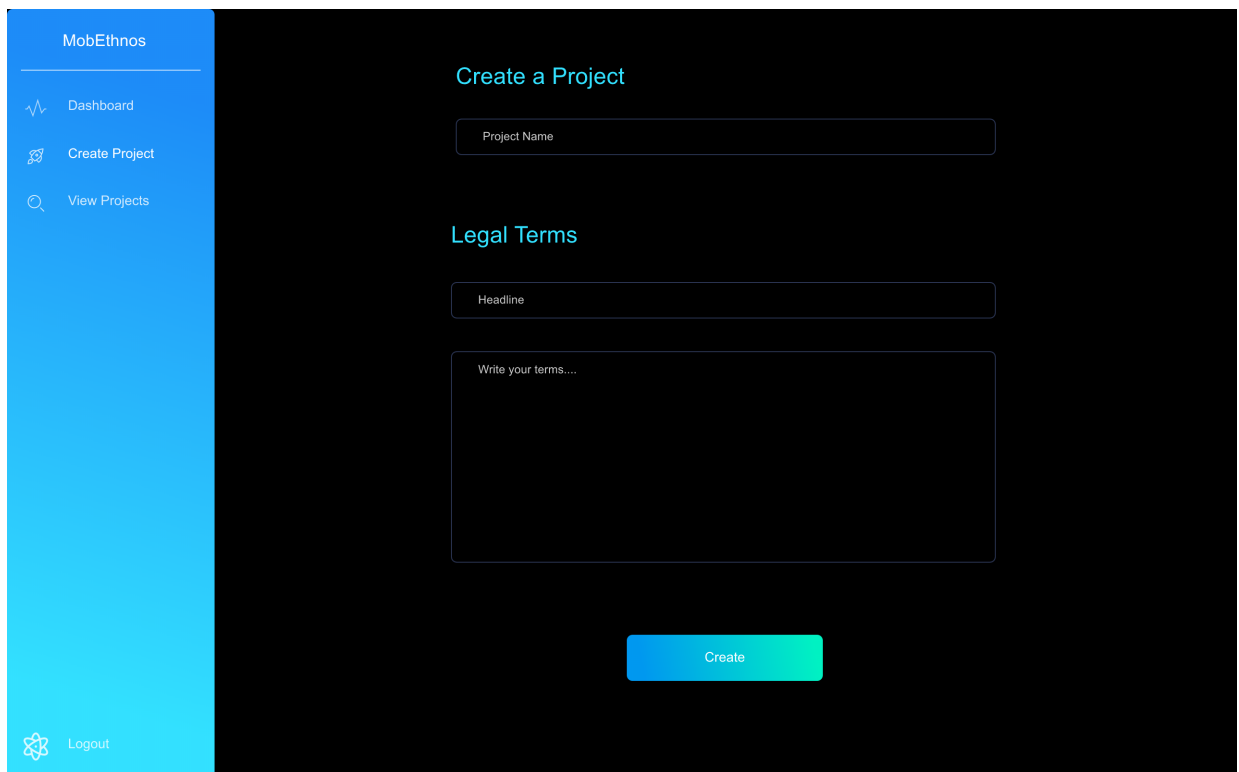


Figure 3.10: Create Project User Interface

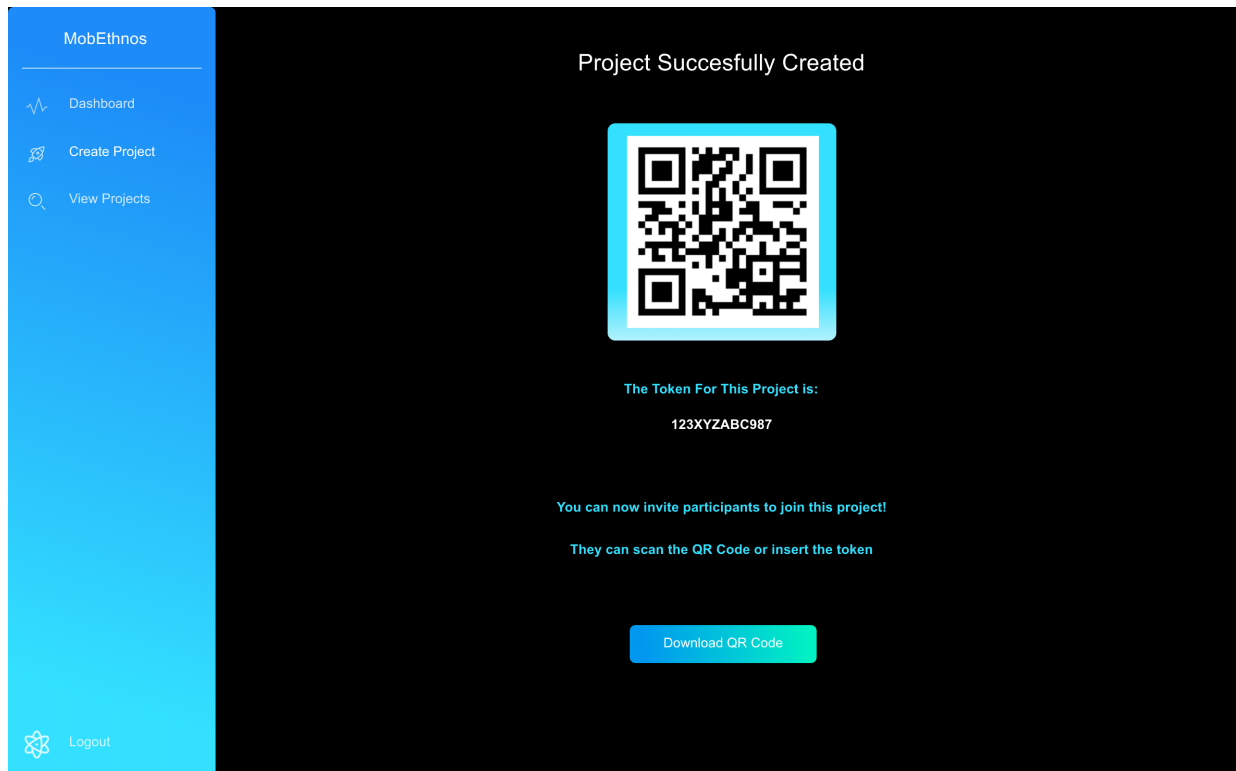


Figure 3.11: Project Created User Interface

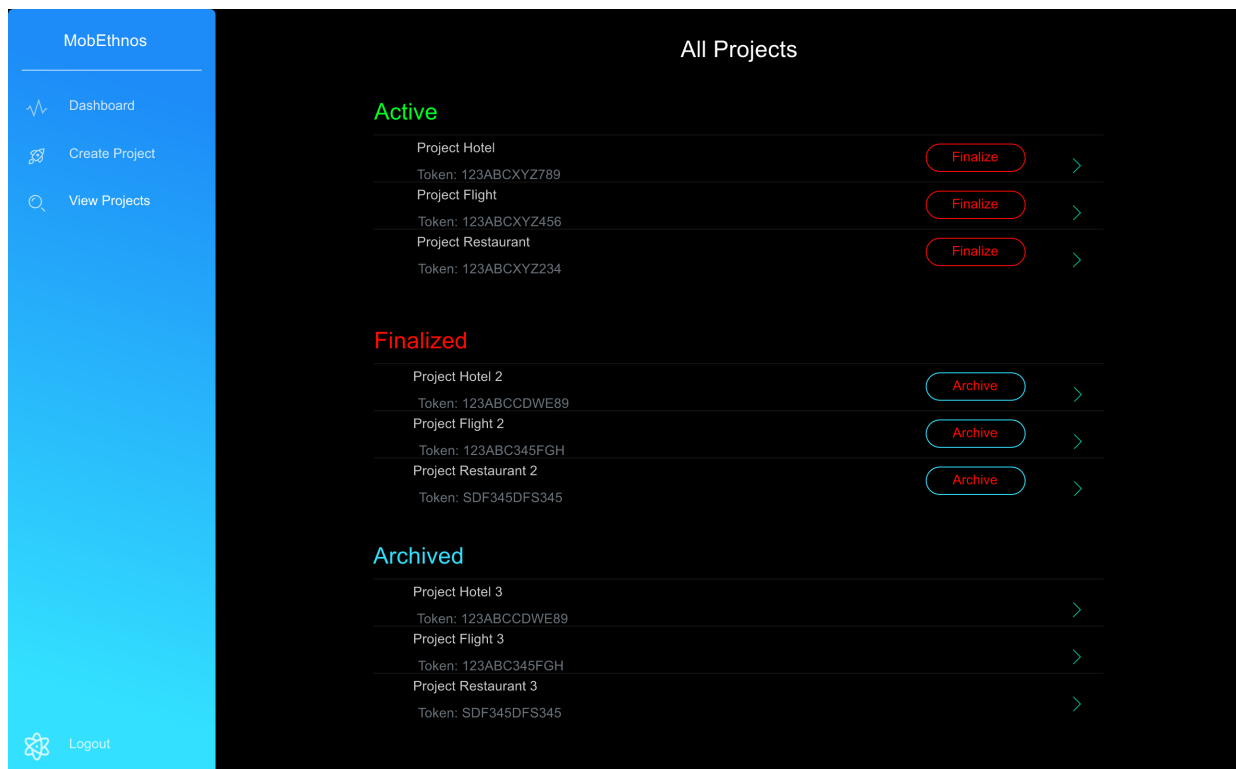


Figure 3.12: View Projects User Interface

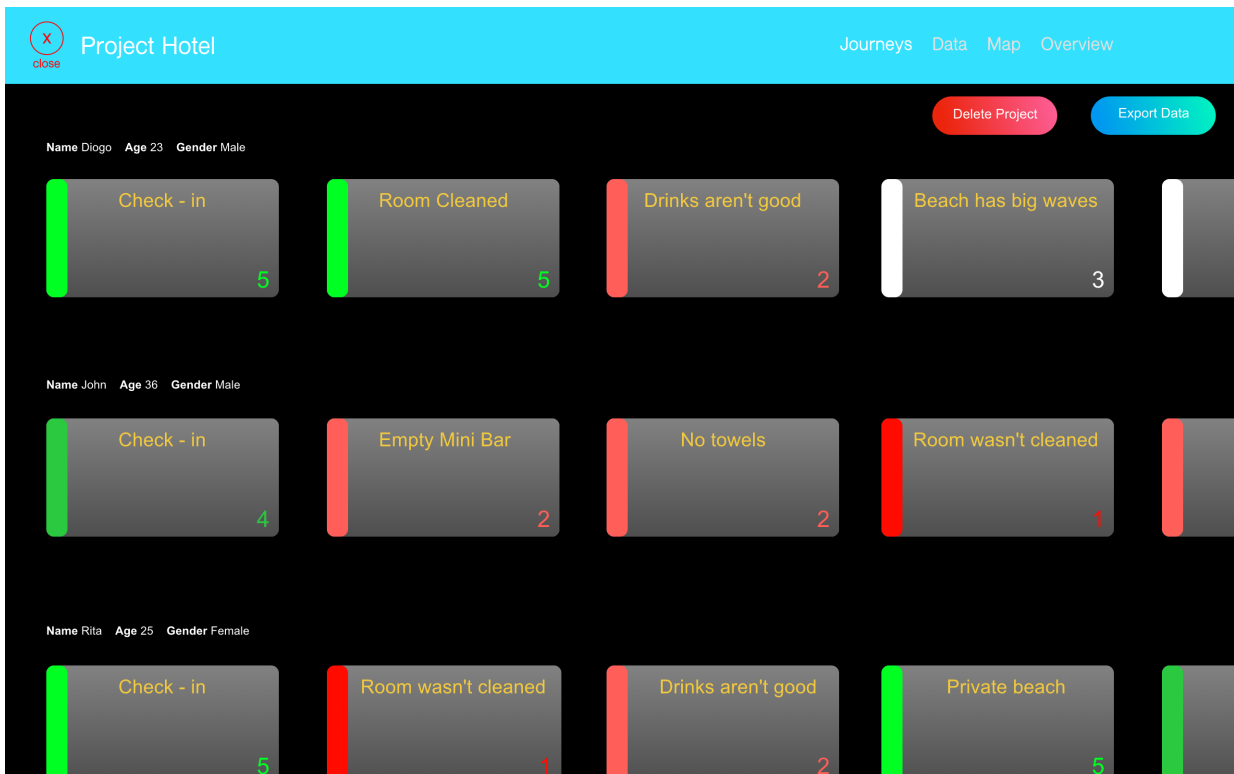


Figure 3.13: Project Journeys User Interface

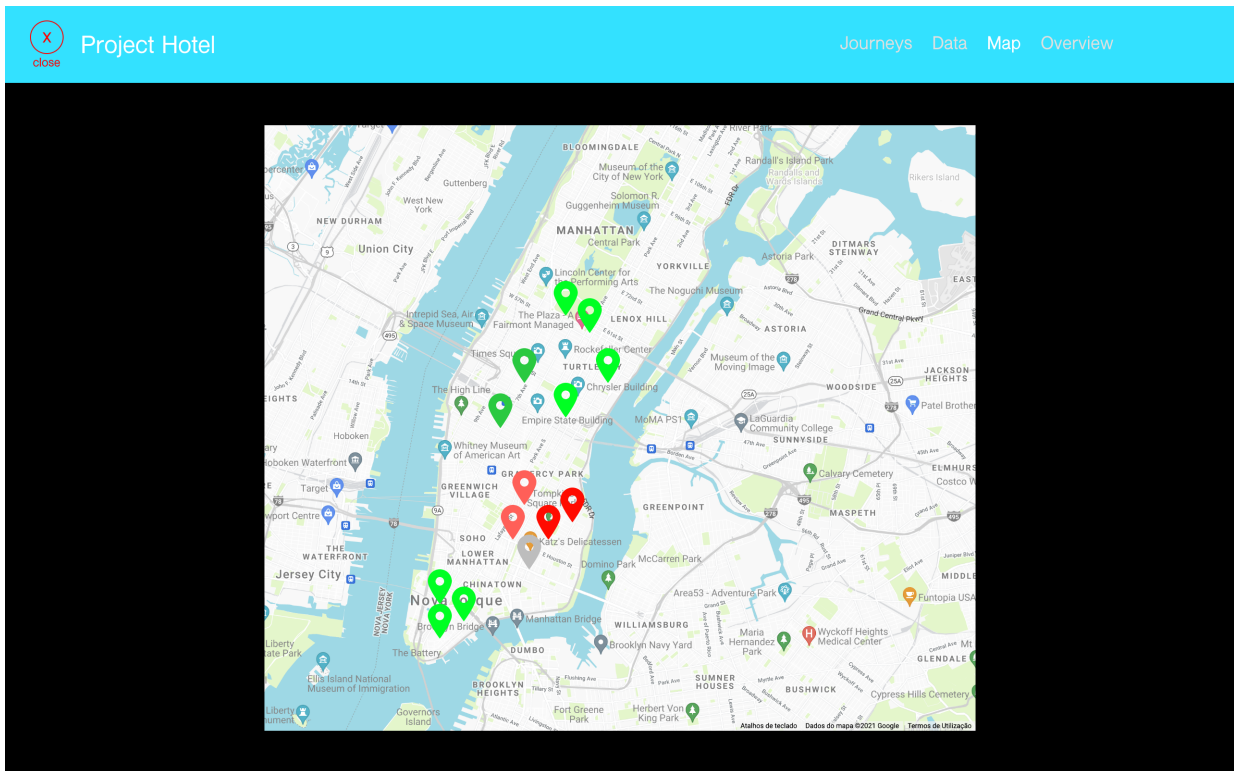


Figure 3.14: Project Map User Interface

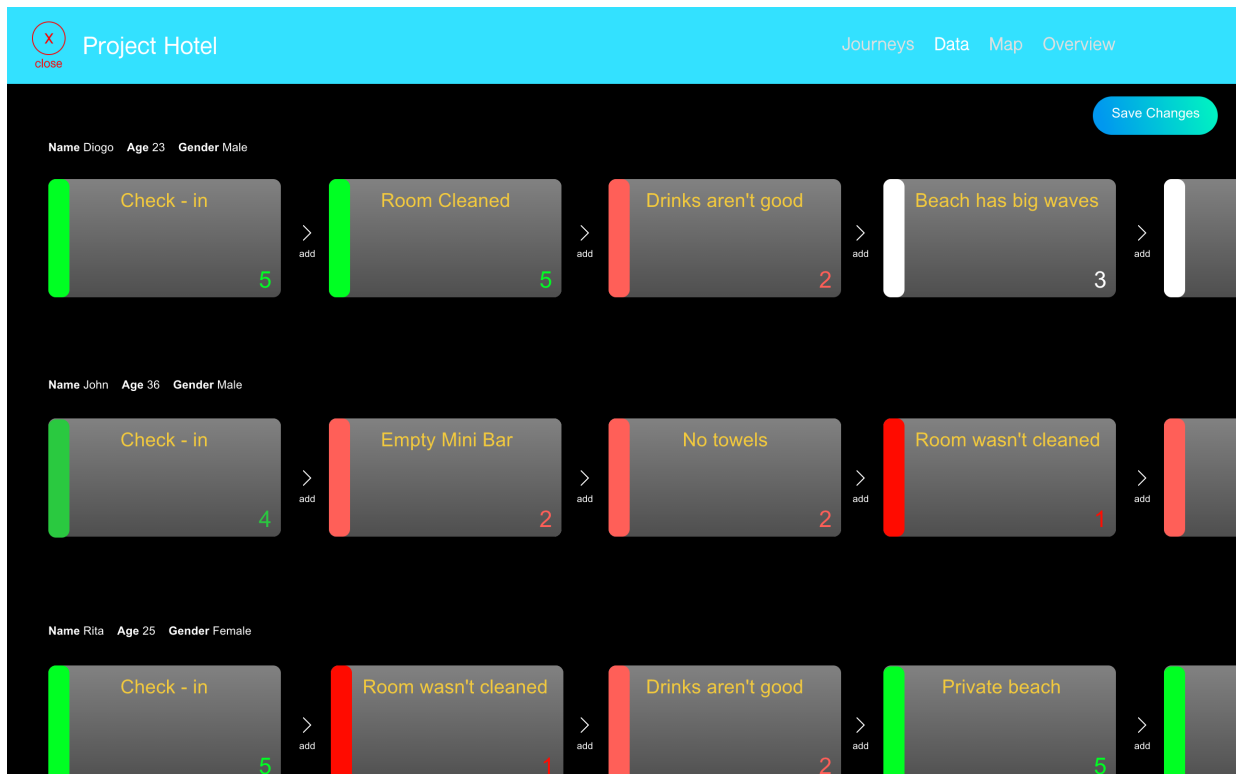


Figure 3.15: Project Data User Interface

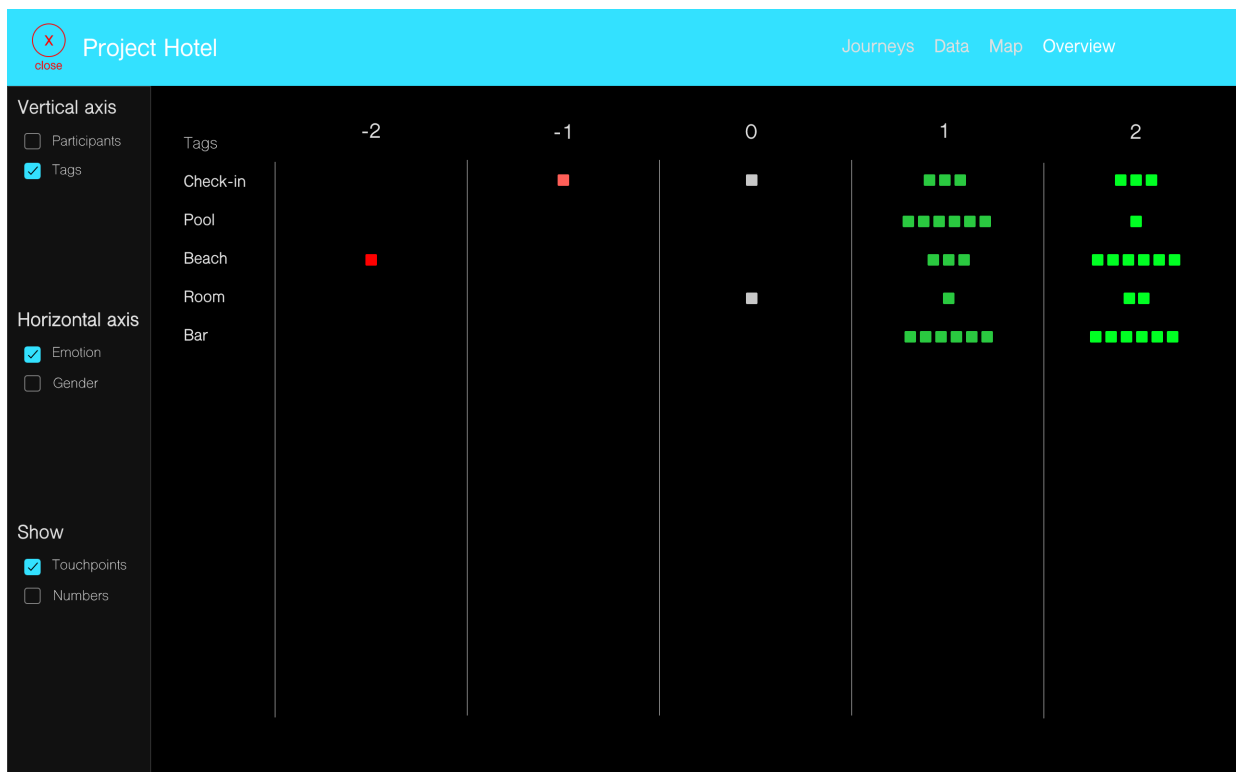


Figure 3.16: Project Overview User Interface

Mobile Application

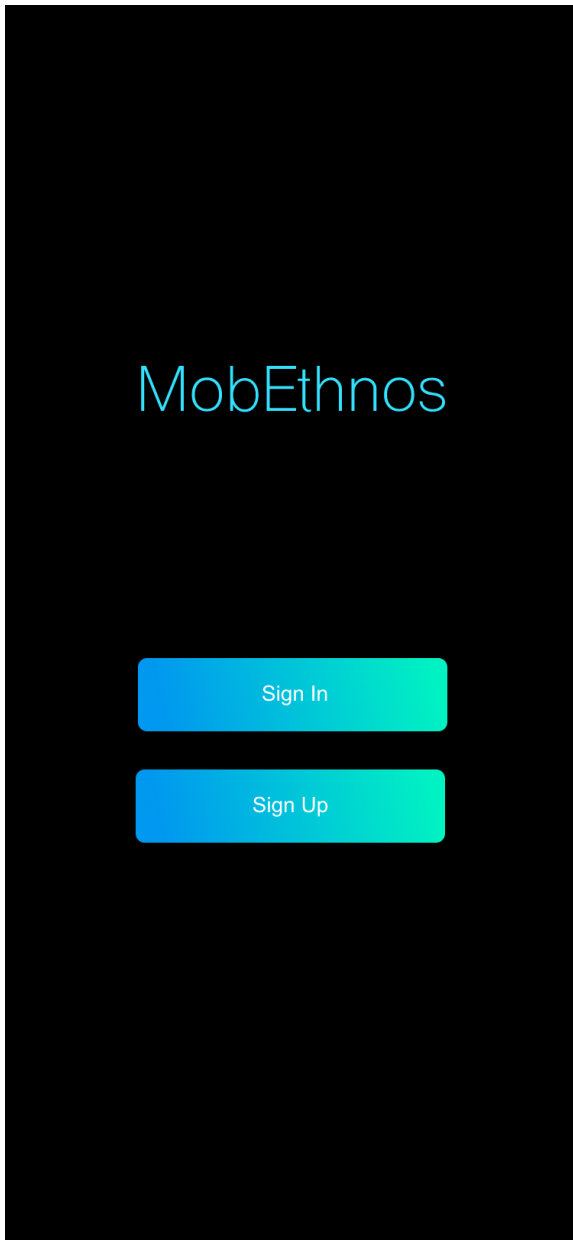


Figure 3.17: Homepage User Interface.

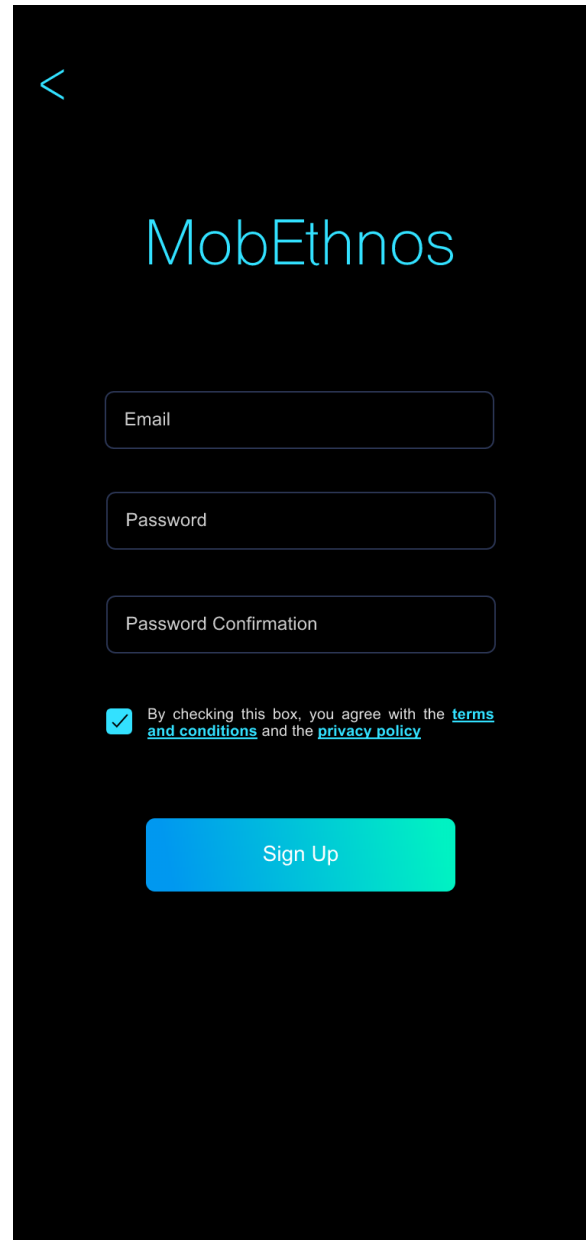


Figure 3.18: Sign Up User Interface.

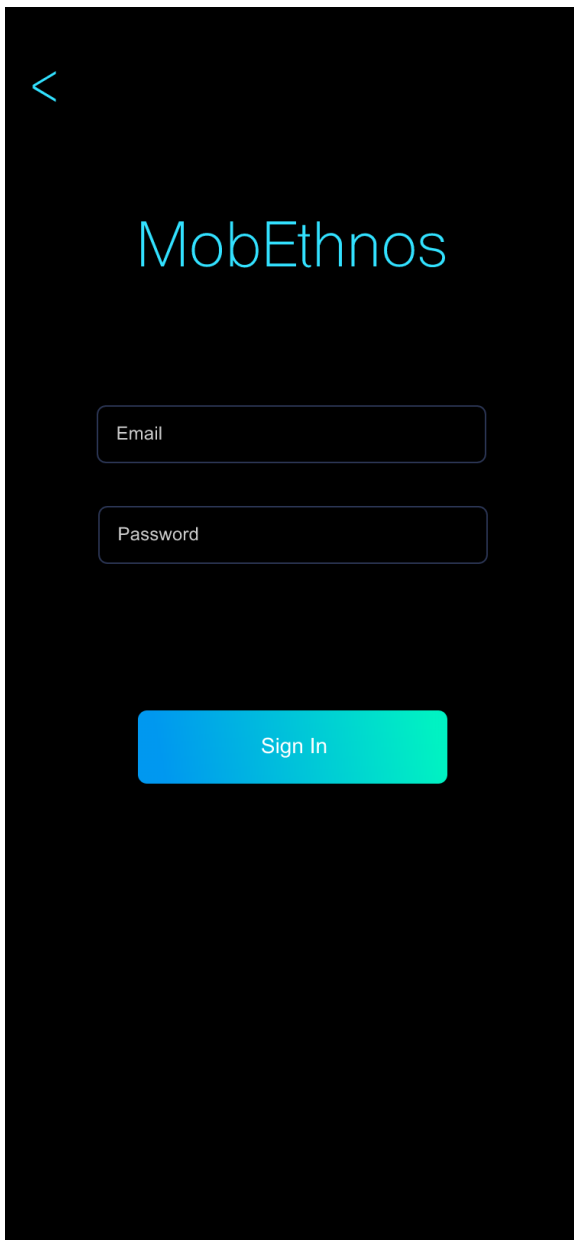


Figure 3.19: Sign In Interface.

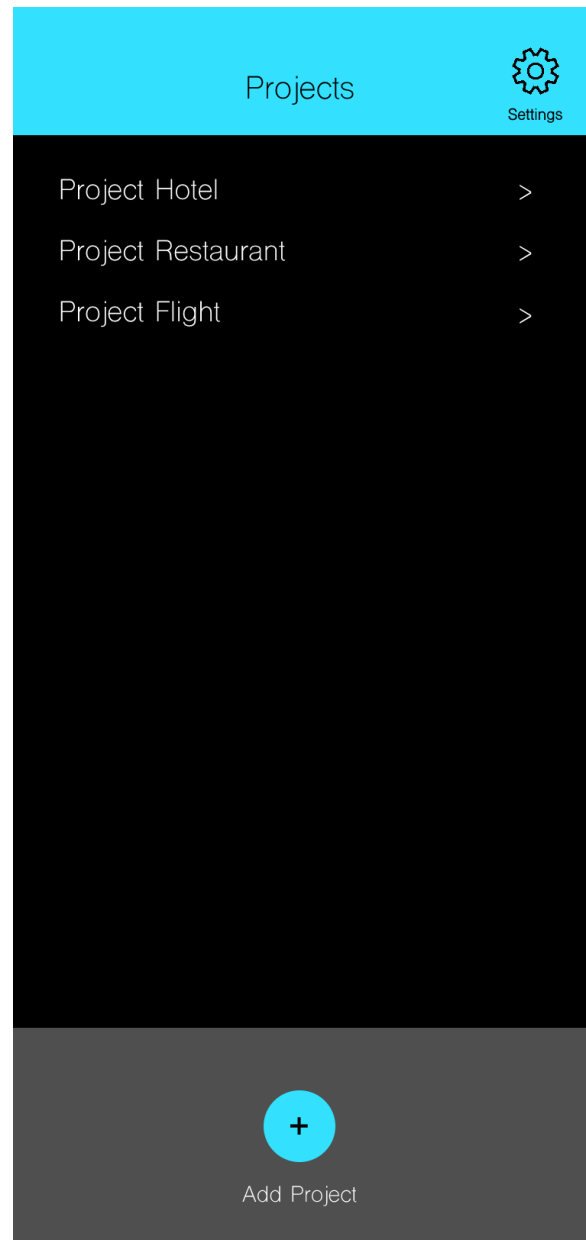


Figure 3.20: Choose Project User Interface.

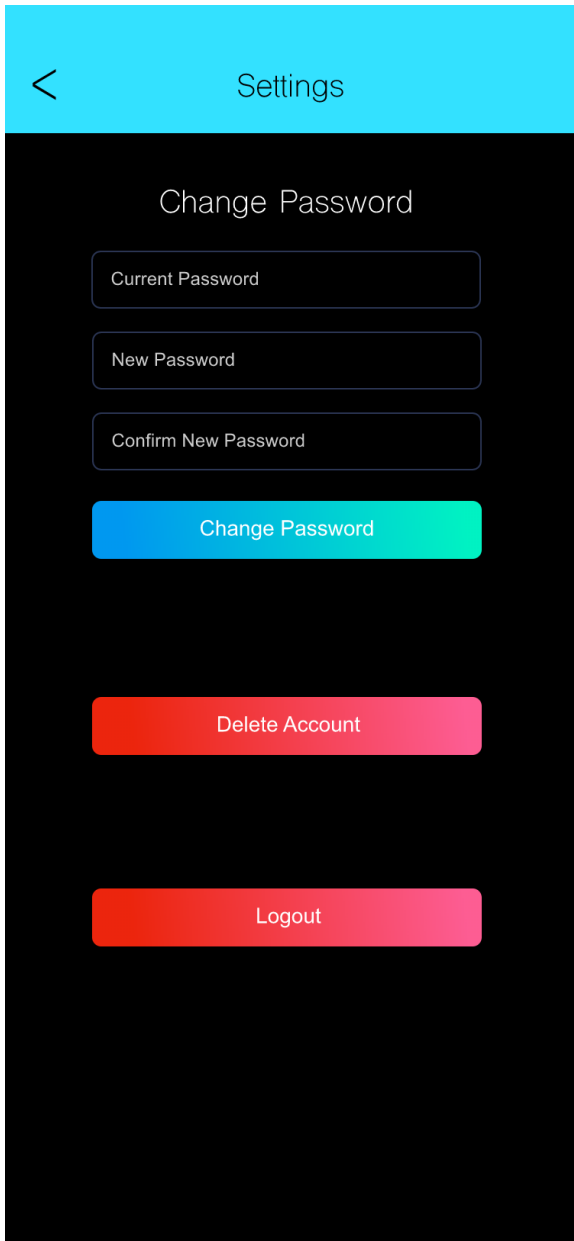


Figure 3.21: Settings Interface.

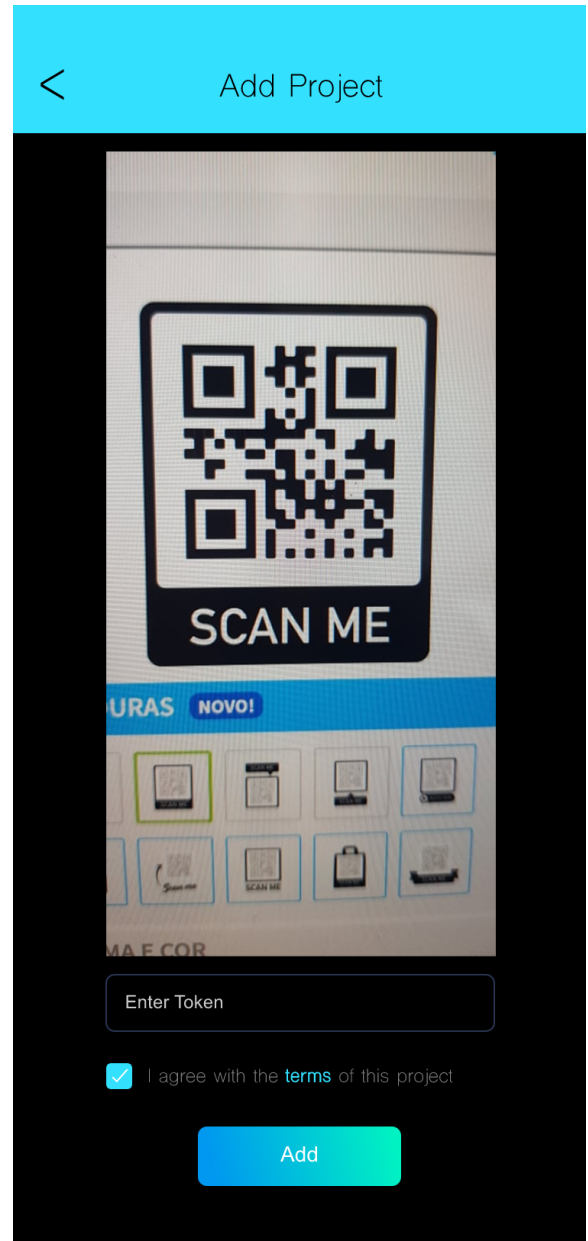


Figure 3.22: Add Project User Interface.

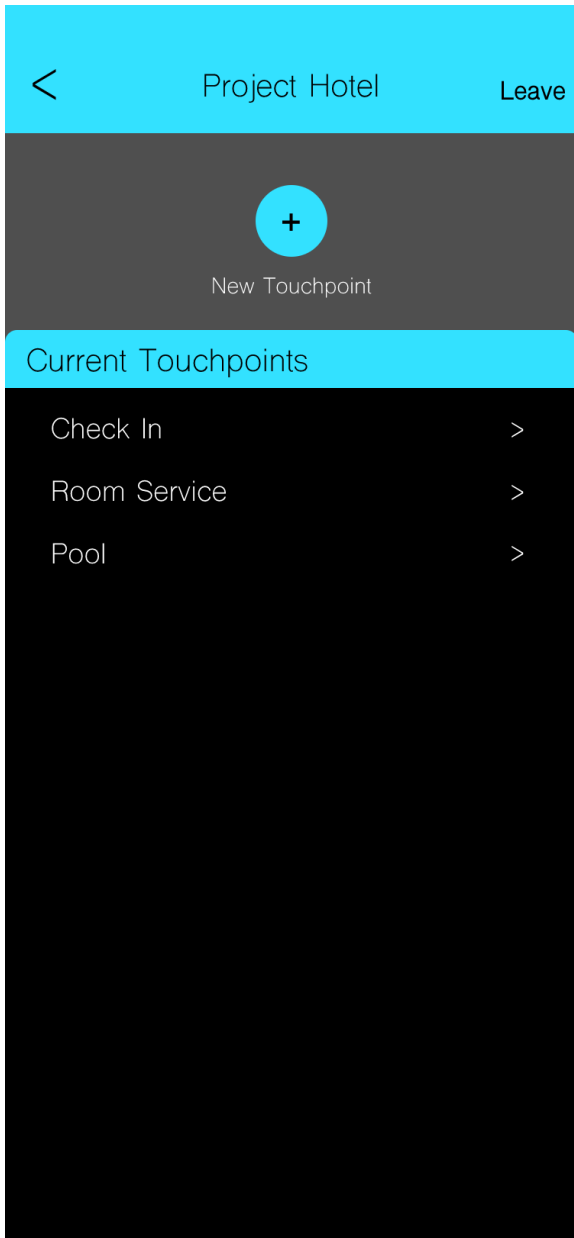


Figure 3.23: Project User Interface.

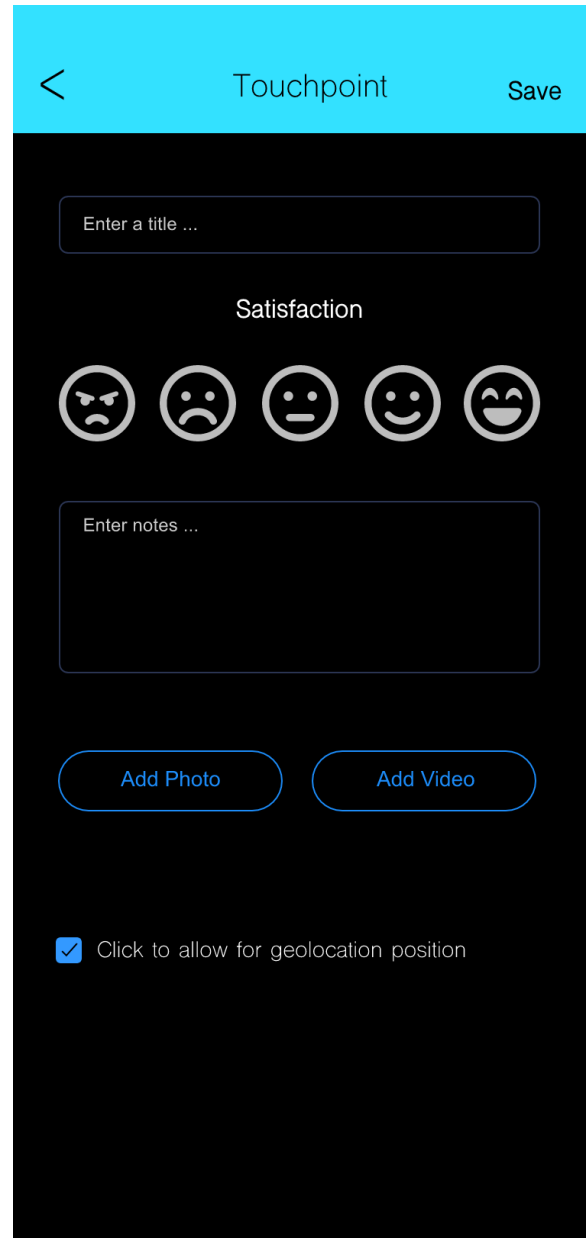


Figure 3.24: New/Edit Touchpoint User Interface.

3.3 Non Functional Requirements

The Non Functional Requirements of this project should provide the ability to make this software tool fast and intuitive to use.

The first and most important non-functional requirement should be usability. It consists of having a simple to use User Interface. The UI was made with this in mind. Also, the researcher requirements will have a User Manual to help explain all the procedures to use all the functionalities of the website in a simpler way.

Another non-functional requirement is scalability. With this requirement, we do not want to have a decrease in performance by having a class of 200 students using the application at the same time. Firebase already provides scalability, their database is indexed in a way to not have drops in performance.

Availability is also important. Deferred Syncing is already automatically implemented in the Firebase software development kit and provides availability, allowing participants to use the application even when an Internet connection is not available.

3.4 Requirements List

This chapter will present a list of requirements for the participant and researcher. It will contain the name of the requirement, the use case ID, and the MoSCoW prioritization method for each one.

MoSCoW prioritisation divides the requirements into four specific priorities.

- **Must Have** - Mandatory requirements needed for the project to work.
- **Should Have** - Not mandatory, but adds relevant value to the project.
- **Could Have** - Nice to have, but if not implemented will have a small impact on the project.
- **Will Not Have** - Not to be implemented at this specific time, but can be implemented in the future.

3.4.1 Participant Requirements

1. **Sign Up** - ID: 01 - Must Have
2. **Sign In** - ID: 03 - Must Have
3. **Join a New Project** - ID: 05 - Must Have
4. **View Projects that you are Involved in** - ID: 06 - Must Have
5. **Change Password** - ID: 07 - Must Have
6. **Delete Account** - ID: 08 - Must Have
7. **View all Touchpoints Submitted** - ID: 09 - Must Have
8. **Edit Touchpoint** - ID: 10 - Must Have
9. **Add New Touchpoint** - ID: 11 - Must Have
10. **Leave Project** - ID: 12 - Must Have
11. **Logout** - ID: 13 - Must Have

3.4.2 Researcher Requirements

1. **Sign Up** - ID: 02 - Must Have
2. **Sign In** - ID: 04 - Must Have
3. **Change Password** - ID: 14 - Must Have
4. **Delete Account** - ID: 15 - Must Have
5. **Create Project** - ID: 16 - Must Have
6. **View Own Projects** - ID: 17 - Must Have
7. **Finalize Projects** - ID: 18 - Should Have
8. **Archive Projects** - ID: 19 - Should Have
9. **View Project Journeys** - ID: 20 - Must Have
10. **Export Data** - ID: 21 - Could Have
11. **Delete Project** - ID: 23 - Must Have

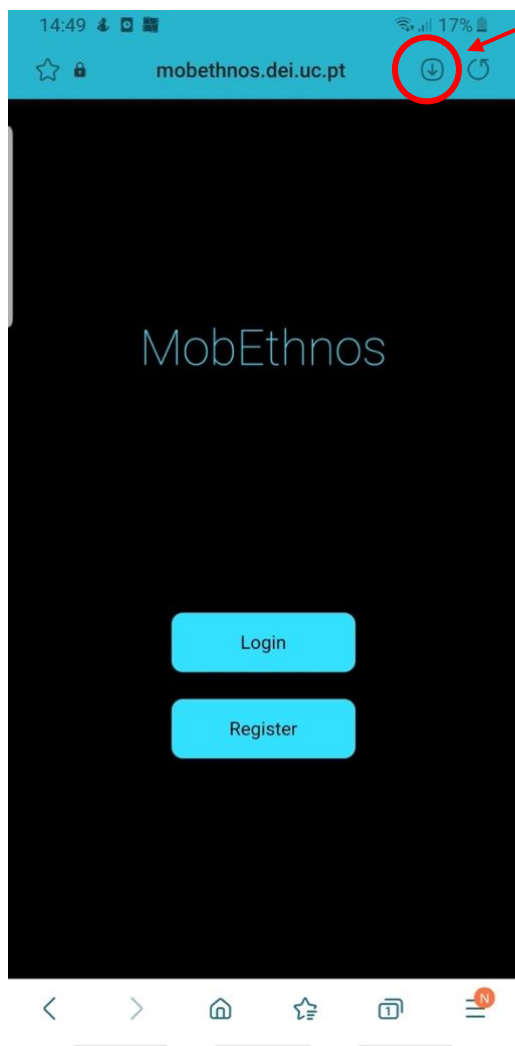
12. **Manipulate Project Journeys** - ID: 22 - Must Have
13. **Show Project Map** - ID: 24 - Should Have
14. **Show Overview Graph** - ID: 25 - Could Have
15. **Logout** - ID: 26 - Must Have

This page is intentionally left blank.

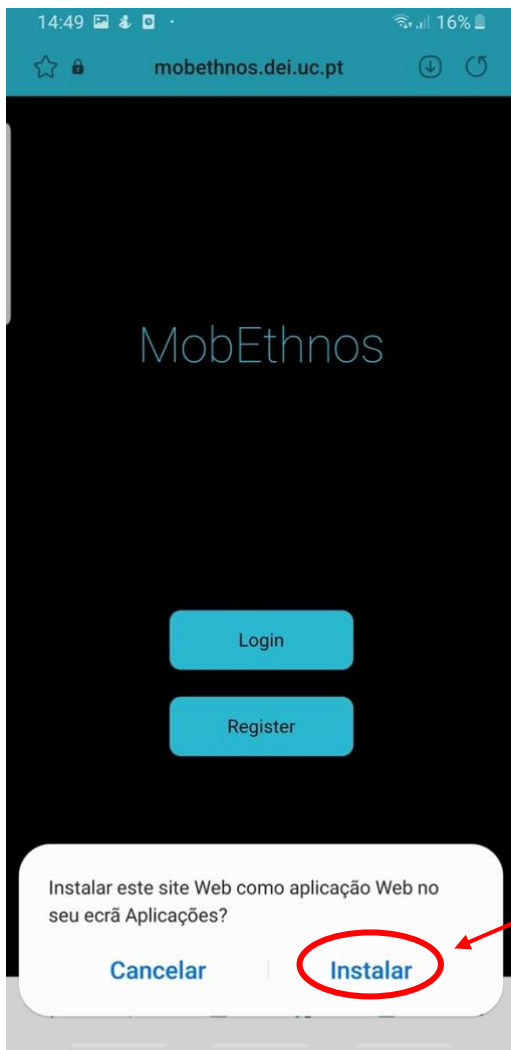
Guia de instalação da Aplicação Web Progressiva

Android

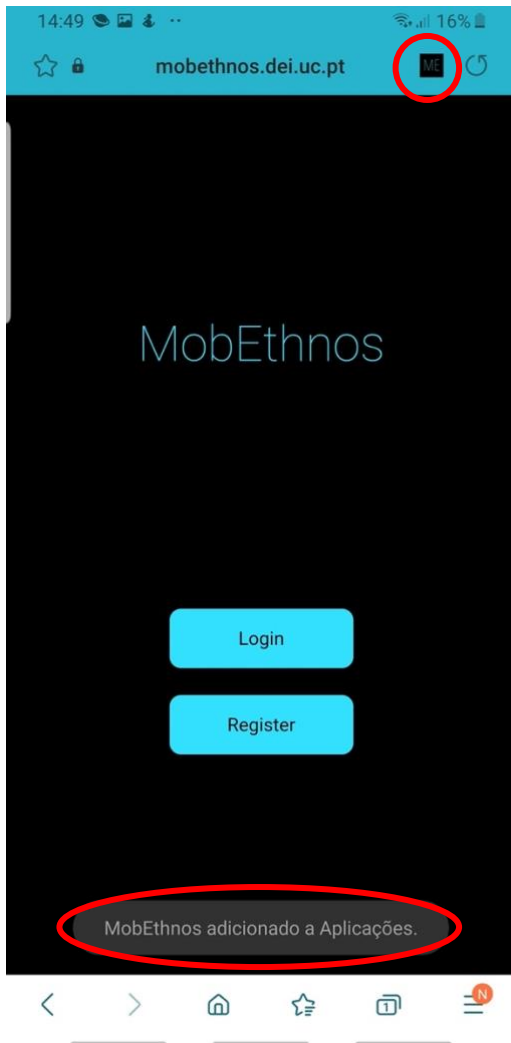
1. Aceder ao website mobethnos.dei.uc.pt
2. Carregar no botão para instalação



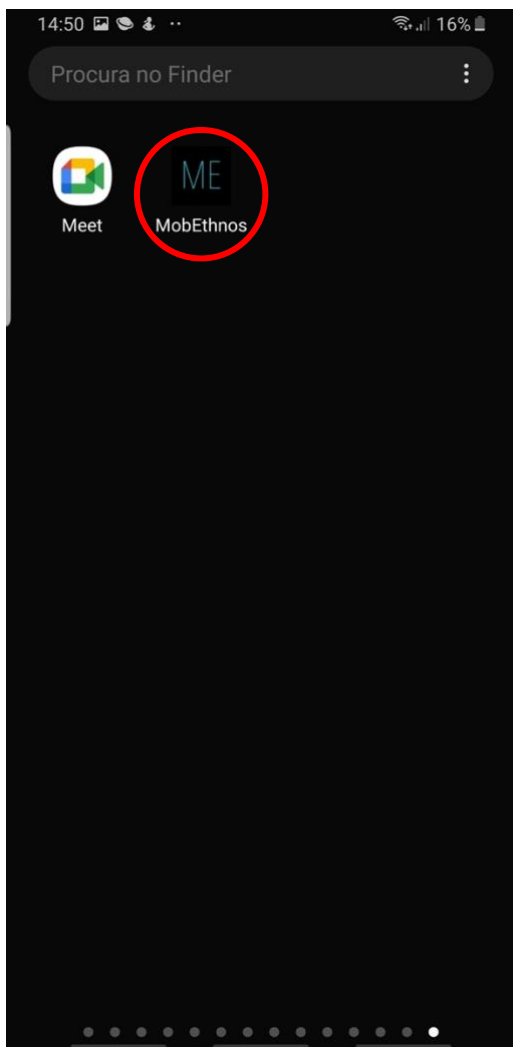
3. Carregar no botão instalar



4. Deve aparecer a seguinte linha a dizer “MobEthnos adicionado a Aplicações” e em vez de ter um botão para instalar, deve aparecer o ícone a da aplicação

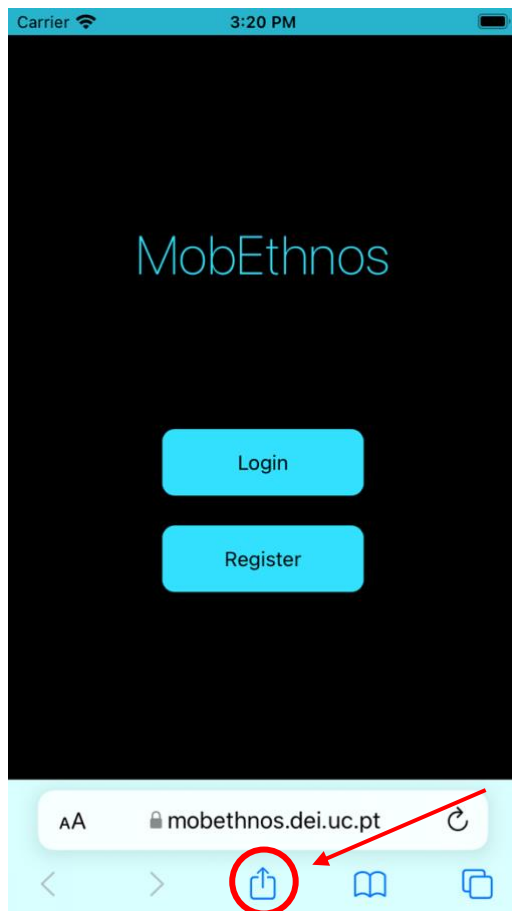


5. Depois é só aceder ao menu das Aplicações e utilizar como se fosse uma aplicação móvel nativa

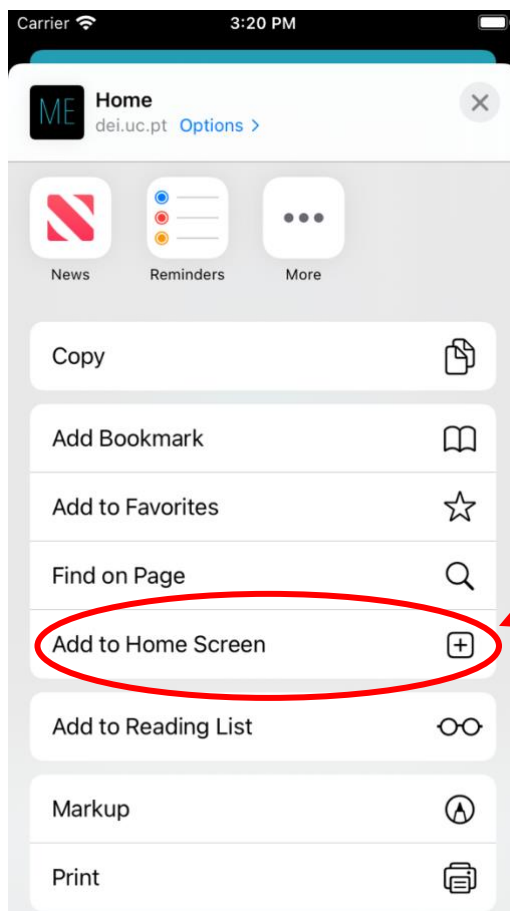


iOS

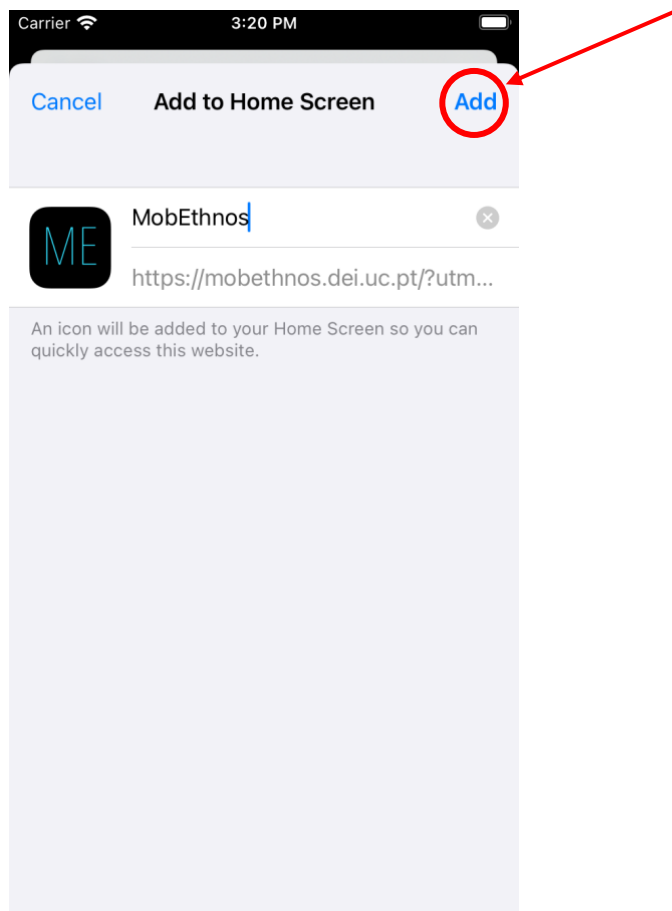
1. Aceder ao website mobethnos.dei.uc.pt por via do Safari
2. Carregar no botão "Share"



3. Carregar no botão “Add to Homescreen”



4. Carregar no botão "Add"



5. A aplicação deverá aparecer no Ecrã Inicial

