NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

APPLICATIONS ACROSS CO-LOCATED DEVICES

STUDYING A FRAMEWORK-DRIVEN DESIGN APPROACH

PEDRO EMANUEL ALBUQUERQUE E BAPTISTA DOS SANTOS

Master in Computer Science

DOCTORATE IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2021

DEPARTMENT OF
COMPUTER SCIENCE

APPLICATIONS ACROSS CO-LOCATED DEVICES

STUDYING A FRAMEWORK-DRIVEN DESIGN APPROACH

PEDRO EMANUEL ALBUQUERQUE E BAPTISTA DOS SANTOS

Master in Computer Science

Adviser: Rui Miguel Neves Gonçalves Madeira
Assistant Professor, Setúbal School of Technology, Polytechnic Institute of Setúbal

Co-adviser: Nuno Manuel Robalo Correia
Full Professor, NOVA School of Science and Technology, NOVA University Lisbon

Examination Committee

Chair: Pedro Manuel Corrêa Calvente Barahona
Full Professor, NOVA School of Science and Technology, NOVA University Lisbon

Rapporteurs: Johannes Schöning
Full Professor, School of Computer Science, University of St.Gallen

Luís Manuel Pinto da Rocha Afonso Carriço
Full Professor, Faculty of Sciences, University of Lisbon

Adviser: Rui Miguel Neves Gonçalves Madeira
Assistant Professor, Setúbal School of Technology, Polytechnic Institute of Setúbal

Members: Rui João Peixoto José
Associate Professor with Tenure, School of Engineering, University of Minho

Teresa Isabel Lopes Romão
Associate Professor, NOVA School of Science and Technology, NOVA University Lisbon

Rui Pedro da Silva Nóbrega
Assistant Professor, NOVA School of Science and Technology, NOVA University Lisbon

DOCTORATE IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2021

Applications across Co-located Devices

This document was created with the (pdf/Xe/Lua)LATEX processor and the NOVAthesis template (v6.9.10) [Lou21].

À minha mãe, à minha avó Tita e ao meu avô Ilídio.

Acknowledgements

I would like to thank my advisors, Prof. Rui Neves Madeira and Prof. Nuno Correia, for all of the good advises and guidance during my Ph.D. In particular, the close relationship and supervision provided by Prof. Rui Neves Madeira greatly contributed to the success of this work. I would also like to thank the other students and researchers at the *NOVA-LINCS Multimodal Systems Group* for all of the suggestions and help testing out some ideas and prototypes.

I would like to thank to the *NOVA LINCS* and the *Department of Computer Science* of the *NOVA School of Science and Technology/NOVA University Lisbon* for providing the conditions required to conduct by research. I would also like to thank to *FCT (Foundation for Science and Technology)* for the Ph.D. scholarship that they granted me to conduct my research (SFRH/BD/96899/2013).

Finally, I need to thank my family for all the support during this important stage of my personal and academic life. I am especially grateful for all the love and comprehension of my mother who has been always besides me when I needed her. Without her this thesis would surely not have been possible. My maternal grandmother and my maternal grandfather have also always supported me. I am only sad that they have passed away over the course of my Ph.D. and that they are no longer here to see me finish it. I know that they would be proud.

Abstract

We live surrounded by many computing devices. However, their presence has yet to be fully explored to create a richer ubiquitous computing environment. There is an opportunity to take better advantage of those devices by combining them into a unified user experience. To realize this vision, we studied and explored the use of a framework, which provides the tools and abstractions needed to develop applications that distribute UI components across co-located devices.

The framework comprises the following components: authentication and authorization services; a broker to sync information across multiple application instances; background services that gather the capabilities of the devices; and a library to integrate web applications with the broker, determine which components to show based on UI requirements and device capabilities, and that provides custom elements to manage the distribution of the UI components and the multiple application states. Collaboration between users is supported by sharing application states. An indoor positioning solution had to be developed in order to determine when devices are close to each other to trigger the automatic redistribution of UI components.

The research questions that we set out to respond are presented along with the contributions that have been produced. Those contributions include a framework for cross-device applications, an indoor positioning solution for pervasive indoor environments, prototypes, end-user studies and developer focused evaluation. To contextualize our research, we studied previous research work about cross-device applications, proxemic interactions and indoor positioning systems.

We presented four application prototypes. The first three were used to perform studies to evaluate the user experience. The last one was used to study the developer experience provided by the framework. The results were largely positive with users showing preference towards using multiple devices under some circumstances. Developers were also able to grasp the concepts provided by the framework relatively well.

Keywords: cross-device applications, context-awareness, application framework, proxemics, indoor positioning, user experience, human-computer interaction

Resumo

Vivemos rodeados de dispositivos computacionais. No entanto, ainda não tiramos partido da sua presença para criar ambientes de computação ubíqua mais ricos. Existe uma oportunidade de combiná-los para criar uma experiência de utilizador unificada. Para realizar esta visão, estudámos e explorámos a utilização de uma *framework* que forneça ferramentas e abstrações que permitam o desenvolvimento de aplicações que distribuem os componentes da interface do utilizador por dispositivos co-localizados.

A *framework* é composta por: serviços de autenticação e autorização; *broker* que sincroniza informação entre várias instâncias da aplicação; serviços que reúnem as capacidades dos dispositivos; e uma biblioteca para integrar aplicações *web* com o *broker*, determinar as componentes a mostrar com base nos requisitos da interface e nas capacidades dos dispositivos, e que disponibiliza elementos para gerir a distribuição dos componentes da interface e dos estados de aplicação. A colaboração entre utilizadores é suportada através da partilha dos estados de aplicação. Foi necessário desenvolver um sistema de posicionamento em interiores para determinar quando é que os dispositivos estão perto uns dos outros para despoletar a redistribuição automática dos componentes da interface.

As questões de investigação inicialmente colocadas são apresentadas juntamente com as contribuições que foram produzidas. Essas contribuições incluem uma *framework* para aplicações multi-dispositivo, uma solução de posicionamento em interiores para computação ubíqua, protótipos, estudos com utilizadores finais e avaliação com programadores. Para contextualizar a nossa investigação, estudámos trabalhos anteriores sobre aplicações multi-dispositivo, interação proxémica e sistemas de posicionamento em interiores.

Apresentámos quatro aplicações protótipo. As primeiras três foram utilizadas para avaliar a experiência de utilização. A última foi utilizada para estudar a experiência de desenvolvimento com a *framework*. Os resultados foram geralmente positivos, com os utilizadores a preferirem utilizar múltiplos dispositivos em certas circunstâncias. Os programadores também foram capazes de compreender a framework relativamente bem.

Palavras-chave: aplicações multi-dispositivo, contexto computacional, *framework*, proxémica, posicionamento em interiores, experiência de utilizador, interação pessoa-máquina

Contents

Appendices

List of Figures

List of Tables

List of Listings

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called main.gls) hasn't been created.

Check the contents of the file main.glo. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like \gls or \glsadd) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add nomain to your package option list when you load glossaries-extra.sty. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add automake to your package option list when you load glossaries-extra.sty. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "main"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "main"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

Acronyms

This document is incomplete. The external file associated with the glossary 'acronym' (which should be called `main.acr`) hasn't been created.

Check the contents of the file `main.acn`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "main"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "main"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

Symbols

This document is incomplete. The external file associated with the glossary 'symbols' (which should be called `main.sls`) hasn't been created.

Check the contents of the file `main.slo`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "main"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "main"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

1

Introduction

We are "living in a ubiquitous world" as we face widespread ubiquitous computing (Ubicomp) due to the billions of devices shaping the very fabric of an active world [Rod08], but the vision of a truly integrated ubiquitous computing (UbiComp) environment envisioned by Mark Weiser [Wei91] has not yet been fully achieved. From the traditional desktop and laptop PCs to smartphones, tablets, smartwatches, smart TVs, and devices embedded into our houses and cars, there is no lack of opportunities to build integrated experiences [Eur18; Kan19].

However, applications are usually confined to run on a single device at any given time. At best, they are capable of synchronizing certain types of data; and it is common that such synchronization needs to be explicitly turned on [DP08; Gal+16; SW13]. Therefore, the main objective of this work is to explore how to take better advantage of the many devices around us by coming up with ways of integrating and combining them.

It should be possible to leverage the strengths of some devices, while also minimizing weaknesses of others. So it is only natural to explore the possibility of building applications that have their UI seamlessly and pervasively distributed across multiple co-located devices. Such unified experience opens new opportunities to develop rich and integrated user experiences, which may contribute to higher levels of productivity and user satisfaction.

Given the lack of tools and guidelines required to build this new type of applications, one of the main objectives of this research work was to create them so that developers can take advantage of the increasing pervasiveness of digital devices on our lives. Besides that, the assessment of whether these new applications with deeper integration and interconnectedness are actually beneficial to the potential users was also of the utmost importance.

1.1 Motivation

As can be seen in Figure 1.1, there is a large variety of devices that are commonly used by people. This is all thanks to the explosion in the amount of available computing devices that we have seen in the last few decades.

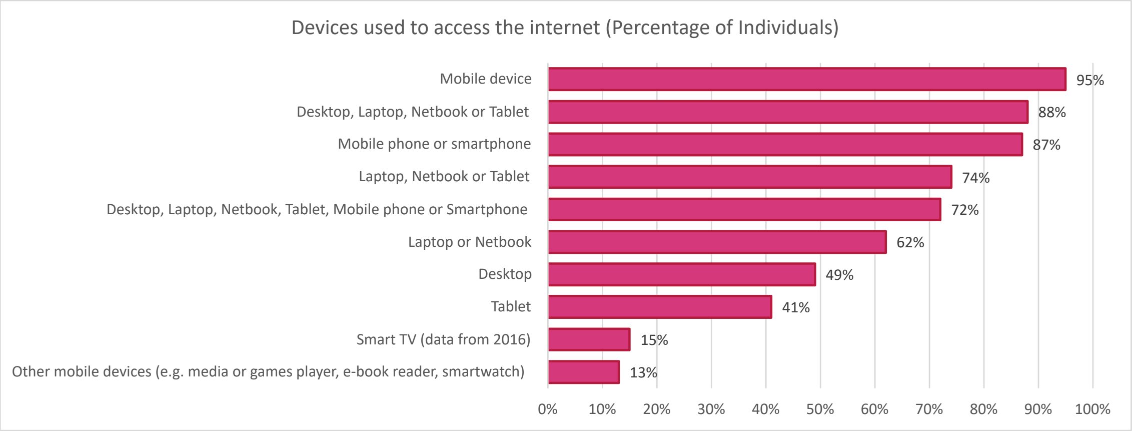


Figure 1.1: Devices used by individuals to access the internet in 2018 in the European Union (it still includes the United Kingdom) according to the *Eurostat* [Eur18]

Desktop computers became commonplace during the 1990s, and laptop computers followed them soon after in the early 2000s. By that time there were already smartphones, such as *Windows Mobile* devices or others introduced by *BlackBerry* and *Nokia*. However, it was only with the introduction of the *iPhone* in 2007 [DB07], and many *Android* smartphones soon after, that they started to become ubiquitous.

The smartphone boom was soon followed by the reinvention of the tablet computer, as earlier attempts failed to gain traction. This trend started with the launch of the first generation *iPad* in 2010 [HN10], with *Android*-based tablets gaining popularity soon after.

Television has also moved away from the traditional broadcasting media paradigm, with Smart TVs integrating many multimedia and Internet-based functions. Modern TVs provide program guides, a web browser, content suggestion, streaming of previously broadcast programs and media from services such as *YouTube* and *Netflix*.

More recently, many wearable devices have entered the consumer market, including the *Apple Watch* [LB15] and other smartwatches, as well as many fitness trackers. They usually work in tandem with the user's smartphone displaying notifications, collecting sensor data, and as an extension to existing applications. However, these devices have had a tendency to gain more standalone capabilities since their introduction.

There have also been advances in Virtual Reality (VR) with the introduction of headsets such as the *Oculus Rift* and *HTC Vive*. These are often used for gaming and entertainment but can also be used for industrial or medical applications. However, it is still unknown for sure whether VR will be widely adopted by consumers, or if it will remain a niche market [Pet18]. Similarly, Augmented Reality (AR) glasses such as the *Google Glass* [Top13], and Mixed Reality (MR) solutions like the *Microsoft HoloLens* [GW16], have yet to reach widespread adoption.

The Internet of Things (IoT) concept has also grown considerably in the last few years [Gar], with just about anything being interconnected, e.g., various sensors, security systems, cars and home appliances. The physical world becomes one big information system, which is expected to jump from 13.8 billion devices in 2021 to 30.9 billion in 2025 [Vai21]. In fact, by 2025 there should be three times more connected IoT devices than traditional computing devices.

Therefore, given the availability of such a vast array of devices, it seemed advantageous to find a proper way of integrating them into a single cohesive environment. This initial idea took inspiration from previously developed work in the area of situated and public displays [KO13]. In particular, it came from research in system specific efforts dealing with the integration of situated displays and mobile devices [Cli13; Pae+04; SMC13].

However, despite a growing body of research exploring this area, it remains a complex topic. It is usually a system specific effort due to the lack of generalizations for the association between ubiquitous computing applications and display resources [Alt+11; Cli13; Dav+09; Gre+11; Jos+08]. Current applications, are mainly focused on digital signage and advertising content, falling short of their full potential [WS12].

Another initial source of ideas was the concept of second screen applications, which promotes the usage of a secondary device to provide an enhanced viewing experience [Cen+15; Cru+17]. It has been documented that users have spontaneously started to use mobile devices while watching TV to access trivia and engage with other viewers [EMa11; GS14; MJ15; Nie11; YG14]. Users have developed new behaviors and habits. They will often transition between multiple devices while performing a single task, or use more than one device simultaneously for a related or even unrelated activities [Goo12].

In summary, the motivation for this thesis came from the realization that, despite the existence of many diverse computing devices, the vision of a truly integrated ubiquitous computing environment with multiple co-located devices has yet to be realized by currently available products, and it has been only partially explored by previous research work.

1.2 The Vision

The desire to find better ways to integrate the various computing devices that surround us has arisen from the realization of the aforementioned advances in consumer electronics and of how ubiquitous various devices have become. Ideally, multiple devices should be meld together as part of the environment in which humans live.

Thanks to the easy access to many different devices, people have naturally started to multi-screen. There are multiple ways of categorizing multi-screening [Goo12]:

- *Sequential multi-screening* – The user moves from one device to another at different times to accomplish a certain task.

- *Related simultaneous multi-screening* – The activities being performed at each device are related with each other.

- *Unrelated simultaneous multi-screening* – The activities being performed at each device are independent and unrelated.

Unrelated simultaneous multi-screening is relatively trivial. The tasks are already unrelated so they should be completely independent from each other and have no need for any kind of integration. It is an usage pattern that is easily achievable just by using different devices for different tasks at the same time.

Sequential multi-screening is part of the problem that we wanted to solve. It should be easy to perform a task on a certain device and then move on to another device. It is true that cloud-based syncing and applications already make this less painful than it used to be, but it is far from being completely seamless. The transition could be fully automated, specially when moving directly from a currently active device to another currently inactive device.

Related simultaneous multi-screening also needed to be addressed. Performing multiple related tasks on different devices may be interpreted as set of smaller sub-tasks with a common goal. Therefore, it should be advantageous to have an integrated way of dealing with all these sub-tasks spread across multiple devices.

Separate devices should be used to perform a complex task. They should easily communicate with each other and be aware of the presence of other devices in their vicinity. For instance, the application's UI may be decomposed into multiple *elements*. If only a single device is available, all the *elements* will be placed on that device. As more devices become available, the *elements* can be automatically distributed by taking into account the devices' resources, i.e., display, speakers, mouse, keyboard, touchscreen, cameras or sensors. Manual migration of related sub-tasks across devices should also be allowed.

One may go even further and add support to complex cross-device interactions, e.g., drag-and-drop to transfer data items across devices. Cross-application composition to better support more complex tasks that require actions to be taken on multiple applications is also a possibility. Despite being part of our vision for the future, these advanced scenarios are not currently supported by our current work. However, we would surely like to explore them as part of future work beyond the scope of this thesis.

The usage patterns presented so far are only focused on a single user using multiple devices. However, the concept may be further expanded to consider more complex scenarios. For instance, the presence of multiple users with multiple personal devices, or the accessibility to public displays. This opens new possibilities in terms of rich collaborative applications fostering social interactions and increased teamwork productivity.

The following scenarios illustrate potential uses of cross-device applications. They exemplify how gestures, movements, and the relative positioning of devices and users can be used as a new interaction medium to build richer user experiences:

1. Lucy is watching a video on her smartphone the moment she enters the living room and sits on the sofa the video is transferred to the TV screen. The smartphone shows the playback controls, allows her to browse the media library and functions

as full-fledged remote control for the TV. If she gets up, the video will once again continue to playback on her smartphone.

She later heads to her office and the video playback is transferred to her laptop. When she grabs her tablet, the playback controls and media library browsing automatically switches to the tablet and the smartphone becomes inactive.

2. Anna is listening to music on her smartphone and enters her car. The music automatically starts to play on the car stereo and some UI elements from the smartphone app are automatically made available on the car's entertainment system.

3. Joe is reading news on his smartphone. He arrives at home and picks up his tablet. The tablet immediately turns on and starts to show the current selected news article. The smartphone remains available to select other articles but the tablet becomes the screen where the article's body is displayed.

   The article that Joe is reading has an image gallery but he wants to view the images in more detail. To do that, he gets near his large desktop monitor and the gallery automatically starts to display there. The tablet can then be used to move to the next or previous picture.

   When Joe puts his smartphone away, the tablet hints that he can slide the screen from the right side to get to the news list if he desires to select another article for reading.

4. Jane is playing a video game on her smartphone. When she enters the office and gets near a large display the game visuals automatically move to it, but the game controls remain on the smartphone.

   Her friend Judy picks her smartphone and gets near the large display. Jane shares the game session with Judy. Judy then selects the game session on her smartphone. Both friends are now playing the game and having fun together.

   John also wants to play, but he does not want to get up. Jane shares the game session with him anyway. He takes out his tablet and selects the game session. He can then play with Jane and Judy with the controls and game visuals placed on his tablet.

5. *Universidade NOVA de Lisboa* offers an app to guide people through their daily lives on campus. It provides access to campus related news, class timetables, test and exam schedules, weather forecasts, the menus of multiple on campus restaurants, notifications when friends are on campus, traffic and public transport information [SMC13].

   Leonard is checking his timetable on his smartphone, but he is feeling overwhelmed by looking at such a small screen. He approaches one of the public displays on campus. The screen notices his presence and starts to show his timetable. His mobile phone works as a remote control to pan and zoom the displayed content.

> Jack approaches the same display with the smartphone in his pocket. His presence is noticed and the display gets split into two areas, according to their relative positions. Some information may be hidden according to the privacy preferences of the users.

There are surely many more possibilities that could be explored. However, these examples give an overview of what can potentially be achieved based on our research. Notwithstanding, it is important to keep in mind that the main objective is to equip third-party developers with the needed tools to build this kind of applications. In the end, it is up to them to be creative and come up with new and interesting ideas.

1.3 Challenges

Despite the progresses that computing devices have suffered in the last few years, they still run on separate operating systems and platforms. While it is common for a vendor to build the same application for multiple platforms, those apps are usually only built in such a way that data exchange among them is possible. For example, Personal Information Manager (PIM) applications share the same e-mail, calendar and contact information, but lack any integration that may help providing a more pervasive and reactive user experience across co-located devices.

Thanks to the popularity of cross-platform development frameworks and web technologies, applications for different devices end up sharing a large portion of their code base but are still run and managed independently when they are running on a given device. They work as separate entities, although they may share some resources and a common back-end infrastructure.

The real challenge is to create applications that work as a whole across multiple devices, where the user interface state is displayed and managed by multiple devices. For this to happen, sharing a common back end is not enough, although it may be used to keep the distributed frond-end coherently synchronized. This represents a profound paradigm shift on how the UI of applications is built and it is not easily achievable using the currently available tools.

To design, implement and run this new kind of applications a comprehensive framework that provides new models, architectural guidelines and software infrastructure had to be developed. Any attempt that does not follow a standard methodology will probably end up being just another system specific effort that will never be able to integrate more than the few parts that were developed along each other.

There should be a framework that is generic enough to give third-party developers enough freedom to implement their ideas. However, it still needs to address all common issues that may arise while building cross-device applications and guarantee interoperability across different devices, operating systems and applications.

Another problem that must be overcome is that each device is generally unaware of its surroundings. However, the relative position of devices and people is an important form of interaction and association that the proposed system must deal with. As far as the framework is concerned, this should be done in a generic way. Nevertheless, as part of its development, testing and validation a positioning and tracking solution had to be integrated.

This thesis is not directly focused on indoor positioning, but it is an aspect that is required in order to allow to work towards implementing its vision. Since there are no widely available positioning solutions that use the devices already carried by users, the research in this area had to be studied in order to find a solution that can be used as part of the development process and evaluation of the framework. Notwithstanding, future advances in the area will surely enhance the user experience of this kind of applications beyond what can currently be achieved.

Besides the already presented technical problems and issues that must be solved, there is also a human factor that has to be taken into account. It is of the utmost importance to determine of if people are receptive to this new type of applications and if they find them advantageous. One must consider how the new possibilities opened by cross-device applications from a Human-computer Interaction (HCI) point of view come together to actually build applications with good usability and user experience. For instance, new affordances need to be considered so that users understand what new interaction possibilities become available as they move themselves and their devices in the physical world.

Application prototypes were built using the proposed framework with those concerns in mind. A series of user studies were conducted using those prototypes to determine how well-received this new type of application would be. Based on the results of those user studies, we could assess that a more widespread introduction of cross-device applications has a good chance of success since the vast majority of the participants gave very positive feedback to our efforts.

1.4 Research Questions

After eliciting the objectives and challenges, we can now present the main research questions that have arisen from our vision:

- **RQ$_1$** – How to create and provide tools that enable developers to build applications that take advantage of co-located devices?

- **RQ$_2$** – How should the application state and graphical user interface be seamlessly distributed across co-located devices to support collaborative environments?

- **RQ$_3$** – How to capture the proxemic relationships between users and devices to associate devices in a cross-device interaction environment?

- **RQ$_4$** – Are applications running across co-located devices beneficial to end-users under certain circumstances?

These research questions can be further summarized into the following research statement: “Propose models and tools to assist developers with the creation of applications which have their UI distributed across multiple co-located devices based on their capabilities and proxemic relationships”.

The models and tools that were proposed as part of our framework tried to encapsulate the details about the behavior of cross-device applications in an way that is easy to use and learn. They are also extensible and modular so that they can be continuously improved. They are also not bound to a specific positioning system because better solutions may become available in the future. Nevertheless, an indoor positioning solution was created and integrated into the framework in order to capture the proxemic relationships allowing us to develop and test the application prototypes used to evaluate the research work.

1.5 Contributions

We aim to realize our vision and to answer our research questions by building upon existing research. As part of the work needed to reach those goals, we expect to produce the following main contributions:

- Framework for Pervasive Cross-device Applications
- Indoor Position System for Pervasive Environments
- Application Prototypes
- User Studies

1.5.1 Framework for Pervasive Cross-device Applications

The development of applications that have their UI distributed seamlessly distributed across multiple devices is the main challenge of this thesis. The UI elements of this type of application should be displayed on multiple devices and their state should automatically kept in sync. This is quite different from what we are used to when dealing with traditional UIs.

Therefore, new abstractions, tools, design principles and architectural guidelines are needed to create applications that follow this paradigm. We hope that the proposal of an application framework encompassing these aspects may bring cross-device applications closer to the mainstream by making their development more accessible. We intend to use existing cross-platform programming languages, libraries and frameworks, building upon them to create our own framework.

As presented in Chapter 2, there have already been some efforts towards the development of cross-device applications, including some generalization efforts with varying degrees of success. There have also been proposals of interactive systems that integrate the proxemic relationship between users and devices.

Nevertheless, our research aims to combine these areas in order to allow the generalization of the development of cross-device applications that incorporate the proxemic relationships of co-located devices, which must be captured using an indoor positioning solution, into their interaction experience. There is also the automatic retrieval of the characteristics of devices to be used as part of the decision system, automatically distributing UI components, which is not seen in other solutions. Therefore, to the best of our knowledge, research on the combination of all these concepts is a novel approach that is worth exploring.

1.5.2 Indoor Position System for Pervasive Environments

Unfortunately, there is still no straightforward way of tracking multiple entities without using complex and expensive solutions, such as motion capture systems [Mar11; Mar+11]. However, one must capture proxemic relationships between users and devices in order to detect across which devices should the UI elements be distributed. Therefore, we had to develop an IPS suitable for this purpose.

From the beginning, it was identified that the IPS should be decoupled from the framework. In fact, it is matter of replacing the existing integration module by another one that targets a different IPSs if developers prefer to use another IPS with the framework.

It was also identified as requirement that the system should take advantage of the capabilities of the devices already carried by the users and the resources already commonly present in home and office environments as much as possible, keeping any additional requirements and infrastructure to a minimum. Therefore, we focused on using Wi-Fi and Bluetooth Low Energy (BLE) which are commonly available technologies.

Despite these requirements, the system had to be capable of determining the proxemic relationships between devices with an acceptable level of accuracy and precision. Latency also had to be adequate in order to provide a good user experience. According to our results of our performance tests and user studies, the system performed well enough given these constraints.

The IPS was developed as a stand-alone system. Since many of its requirements may be shared by other applications, we consider it to be a significant contribution. We hope that it may be used on other projects that need location awareness in pervasive environments and that require a solution based on existing off-the-shelf technologies and devices.

1.5.3 Application Prototypes

As part of the development process we had developed prototypes to test concrete ideas and adjust certain aspects of the framework and of the indoor positioning system (see 5). These application prototypes support different capabilities of the framework depending on when they were developed and what was their main purpose. For instance, the YanuX YouTube Viewer only supports the automatic distribution of UI components. It also does not supports managing and sharing multiple application states. However, JuxtBoard supports all of those features because it was developed at a later stage.

Meanwhile, the YanuX Calculator and YanuX Skeletron are very simple application because the former was focused only on testing the manual distribution of UI components and the management of application states, and the latter was supposed to have a very simple codebase so that it could be easily understood by the participants of a developer focused user study.

The prototypes were used to perform user studies which allowed us to collect valuable feedback and insights. Most of the prototypes, the conducted user studies and the respective results were already presented as part of several published papers. The current exception is the YanuX Skeletron and the corresponding user study with potential developers that is yet to be published as part of a conference's proceedings or a journal.

The prototypes may also serve as an inspiration to other researchers in the area. Some of them may be expanded upon and continue to be developed along with the framework beyond the scope this thesis, possibly leading to further research or to the development of commercial products. Therefore, they should also be considered as a relevant contribution of our work.

1.5.4 User Studies

The final objective was to study if potential users find cross-device applications and the concepts behind our vision as beneficial while we also validate if the framework and the IPS fit the needs of both developers and end-users. We had to determine if applications developed around this cross-device paradigm offer advantages over traditional ones. More importantly, if they do not present significant hindrances in terms of usability and user experience. We also had to check if the abstractions and tools provided to developers were appropriate, thus allowing them to create the applications that they envision. Therefore, we performed several user studies using our application prototypes and developer-focused study.

The usability and user experience provided by our prototypes were be evaluated using HCI evaluation methods. We performed several user studies with voluntary users in a laboratory environment. We took some observations but the participants were also encouraged to think aloud to discuss and give feedback about what they were doing. At the end of each study they were presented with a questionnaire to gather data for further analysis.

We also conducted studies with developers to evaluate how they use the framework to determine how easily they learn to create cross-device applications. We were also interested in assessing if developers acknowledge the framework as being useful and if they find this new type of applications enticing to develop. We followed a similar approach to the user studies performed with end-users, but we followed a more structured approach to target the several aspects of the framework. In general, the results were positive with most participants being able to perform the tasks in an acceptable amount of time. The feedback was also largely positive but there were some suggestions for improvements.

During the evaluation process, both end-users and developers were also exposed to our indoor positioning solution. However, one had to be careful to understand if any issues were a direct consequence of the design of our framework and applications, or if they come from limitations introduced by IPS itself. Nevertheless, as previously mentioned, we also evaluated whether our solution covers the needs of both users and developers.

Since our findings may be of use to other researchers that are interested in cross-device applications, our results are an important part of the publications produced during the course of the thesis and can also be considered a contribution of our work.

1.6 Report Structure

This document presents the work developed in the scope of the PhD thesis that it is part of. It presents the motivation, novelty and relevance behind its vision and the research conducted during its development, implementation and validation. Therefore, it is organized into the following chapters:

1. Introduction - An introductory chapter which presents the motivation behind the research topic covered by this work. It provides an overview into the problems and challenges which should be solved. It also presents the research questions answered by this work along with the contributions it provides.

2. Research Context - This chapter is focused on the study of topics and concepts within fields of study that are related with the subject of the proposed thesis. It covers the relevant technical background and state of the art.

3. Framework for Pervasive Cross-device Applications - This chapter presents in detail the framework that was developed to allow the creation of applications with an UI that spans across multiple devices in a generic and reusable way.

4. Indoor Positioning System for Pervasive Environments - A chapter dedicated to the indoor positioning system that was developed to capture the proxemic relationships established between co-located devices.

5. Application Prototypes - The application prototypes that were developed alongside the framework as part of its development and validation are presented in this chapter.

6. User Studies - The user studies performed with the application prototypes to evaluate the suitability of our solution for both end-users and developers are introduced in this chapter. Their results are also presented and discussed.

7. Conclusions and Future Work - This chapter presents the publications that support the research work done in the scope of this thesis. The answers to the research questions and objectives, along with all addressed problems and challenges, shall be discussed. It also presents conclusions about the research work and ideas for future development.

2

Research Context

It is important to contextualize the research performed as part of this thesis by introducing relevant concepts and the state of the art in relevant areas. Therefore, this chapter is dedicated to the presentation of the following topics:

- Cross-device Applications – Review of existing concepts and some efforts in the area of cross-device applications, including tools that have been previously proposed to ease their development.

- Proxemic Interactions – Research regarding the use of the position and relationships of the entities present in the physical world as an input modality for computers and applications.

- Indoor Positioning Systems – Methods and techniques that can be used as part of an IPS in order to establish the proxemic relationships that will allow seamless cross-device interaction.

2.1 Cross-device Applications

The trend towards a cross-device application ecosystem is something that has been previously identified as the next way forward, based on both device shipment trends and developer interest [TMS14]. We are definitely moving into a new era of multiple device ownership where users should be able to continue doing exactly what they were doing across multiple devices and when they move between them.

This approach should enable new user experiences, such as seamlessly continuing to play the same game on another device, continue watching the same movie or listening to the same song on another device, or just simply displaying the playback controls on one device and the media output on another one, among many other ideas.

Nevertheless, this type of cross-device integrated experience is still not broadly supported, even acknowledging that many of these concepts have been previously discussed (e.g., [Nac+05]) and there have been successful consumer products that offer a glimpse into the future (e.g., *Google Chromecast* and *Apple TV*). Therefore, this section presents previous research efforts that are relevant in the context of this still novel type of applications.

2.1.1 Classification of Cross-device Applications

There is a series of cross-device task patterns that have been previously proposed [Pat20]. These patterns indicate how the tasks are supported by the user interfaces of the platforms involved. Thus, they are defined by the specific relationships that exist between user interfaces, tasks, and platforms. In this context, platform refers to sets of devices with similar interaction resources (e.g., desktops, tablets, smartphones, or smartwatches). The following patterns were identified:

- *Same task, same user interfaces* – The same task is supported through the same UI on the various platforms considered. This is the case when, regardless of the device used, the UI to support the considered task is almost the same. For instance, authentication is usually conducted in the same manner over different platforms.

- *Same task, different user interfaces* – The same task is supported on the various platforms considered but through UIs that differ in some presentation or layout aspect. This usually happens in applications that follow a responsive design approach.

- *Same main task, different subtasks* – The same main task is supported on the various platforms considered, but with some different related subtasks. This refers to when the main task can still be performed on the various platforms, but it has different associated secondary tasks depending on the platform considered. For example, an hotel booking application which requires filling a limited number of fields on the smartphone and on the desktop provides the possibility to enter more details (further secondary tasks).

- *Different, but related, platform-dependent tasks* – Dependencies exist amongst tasks carried out on different platforms and corresponding user interfaces. This means that what is done on one device has an effect on what can be done on another. Thus, some tasks performed on the various platforms may be different but are related to each other. For instance, an application that enables the user to control the video displayed on a TV through a smartphone.

- *Platform-specific task* – These are tasks that are relevant only on specific platforms. This is the case when the tasks depend on specific features available only on certain platforms, for example because they require a lot of screen space or are associated with some mobile positions or specific peripherals (e.g. a camera or a step counter).

In the case of our framework, there is no pattern that is enforced. The only constraint is that the tasks must be part of the same application because multi-application coordination is still not possible. However, within a single application developers are free to implement any of these patterns depending on the requirements of their applications and on which devices are expected to be used for each of the tasks or subtasks.

Based on this classification of tasks which can potentially be supported performed in an cross-device environment, it is important to understand how cross-device user interfaces and applications can be supported in different ways [Pat20]. The simplest form is *sequential application access through different devices* which entails using a single device at a time. This can be achieved either by having the same codebase which adapts to different devices through responsive design, or by having platform specific applications (e.g., the *Gmail* website on a desktop computer and the corresponding application on an *Android* or *iOS* smartphone).

A more complex approach to cross-device applications are *distributed user interfaces* in which more than one device can be used to enter inputs to the application functionality. An example of this approach is to develop an application that shows the navigation and search controls on a smartphone and the currently selected content or query results on a tablet. In this scenario, the roles of the devices are static.

Another interesting concept is that of *migratory user interfaces* that allow users to dynamically move UI components across devices while maintaining their state. As example we can consider the previous scenario for *distributed user interfaces* but in this case users are now able to redistribute components across multiple devices. This means that the roles of the devices can be more dynamic.

Finally, there is *cross-device user interfaces* which keep their components distributed and synchronized across devices. They allow redirecting input events from one device to another and support high-level events obtained by combining events generated in different devices.

Both *distributed* and *cross-device user interfaces* refer to cases in which users access the application through multiple devices at the same time. However, in the case of the latter, there is also a synchronization obtained through event redirection or composition across the devices at the user interface level. Such aspects are not supported in simple *distributed user interfaces*, in which the shared state is only achieved through a shared application back-end.

These categories are not mutually exclusive and there can be a certain level of overlap between them. For instance, in the case of our *framework* we offer support to *sequential application access through different devices*, *distributed user interfaces* and *migratory user interfaces*.

We also incorporate some aspects of *cross-device user interfaces* by promoting the synchronization of the UI through the encapsulation of its global state into an object that is kept in sync with other devices. We also support custom events to be fired and distributed across other devices. One area that our approach is currently lacking is that of the support of high-level events, including support for cross-device interaction gestures.

2.1.2 Conceptual Models

There are multiple conceptual models that were proposed to describe cross-device applications which are interesting to look at and to better frame our research work.

2.1.2.1 Logical Framework by Paternò and Santoro

One of the concept models we looked at is a logical framework proposed by Paternò and Santoro [PS12]. It presents the following ten dimensions that are relevant for the design of this type of systems and applications:

1. **UI Distribution** – This dimension is focused on whether a solution is able to support the distribution of the user interface elements across various devices at a given time. Since there are at least two devices involved in the rendering of the UI, it implies the existence of some coordination supporting the access to the application logic by exploiting input/output from/to the various devices involved in the distribution. An example is when people access large screens to see large amount of information and use a mobile device to enter some queries.

   As for the range of values, the distribution can be *dynamic* when the UI elements can vary their allocation to the devices during a user session, or *static* when the distribution configuration cannot change during a session. For instance, *HuddleLamp* provides support for *dynamic* distribution, but only across devices that are on a table equipped with an overhead depth camera [Räd+14]. However, *Deep Shot* only supports *static* distribution since it is only focused on moving user interface components from one device to another [CL11]. In the case of our framework and developed prototypes, they follow a *dynamic* distribution approach.

2. **UI Migration** – This dimension analyses whether there is some continuity when users change device and still access the same application. Users should be allowed to change the current device in use and then have the application available on a different device while the system automatically preserves the interaction state reached with the first device and offers an adapted UI on the new device.

   The range of values for this dimension is represented by the elements whose state can be preserved and transferred from one device to another. For instance, application data, current state of interactive forms, UI elements, *JavaScript* variables, or session information.

   Various tools support migration only at the level of user interface elements and preserve their state when they move from one device to another. Other like the framework proposed by Ghiani et al. support the migration of session information and JavaScript variables [GPS12]. Early work such as *Pick-and-Drop* was limited to just transferring data between devices. In the case of our framework and prototype applications the state of groups of UI elements defined by developers are kept in sync based on information stored on a server-side component.

3. **UI Granularity** – This dimension considers the granularity of the user interface that is manipulated across various devices through distribution or migration. It can have the following range of values:

    - *Groups of UI elements* – This case considers the possibility of distributing structured parts of user interfaces across various devices. For instance, navigation bars or articulated content areas with text and images.
    - *Single UI elements* – In this case, single UI elements are distributed across devices.
    - *Components of UI elements* – This includes interactive elements which are usually characterized by *prompt*, *input*, and *feedback* are distributed across devices. For example, the user enters an input through a mobile device and the resulting feedback is shown on a large screen.

    As an example, *Deep Shot* considers only migration of the entire current user interface [CL11]. Meanwhile, *Panelrama* focuses on panels, which are groups of elements that are associated with a specific intent [YW14]. There are even cases that allow the selection of specific parts of single elements [FP14; MP16].

    In the case of our framework and prototypes, the UI granularity is defined as *groups of UI elements* elements created by the developers. However, these groups can have implicitly defined roles, such as *prompt*, *input*, and *feedback* in the scope of the application. Therefore, they can also be seen as *components of UI elements*.

4. **Trigger Activation Type** – This dimension analyzes how the request for a change in the cross-device user interface is triggered. This change could then activate a migration or a redistribution of the UI. The simplest case is *user-initiated* which allows the user to actively select when, to which device and what should be changed. We can further distinguish between *push* and *pull* modalities depending on whether the triggered migration is from the current local device to a remote one or vice versa.

    With *automatic trigger* the system autonomously activates the change when it recognizes the verification of suitable contextual conditions. For instance, in case of a high battery consumption level and a user's proximity to another device the system might decide that a device change is appropriate and takes the action of selecting the new device to be used.

    Another option is a *mixed type of trigger activation* which is partially suggested automatically and partially determined by the user. For instance, the system first automatically suggests a change to the user who is still able to modify some parameters in the request.

Some examples of the application of this dimension include the platform presented by Ghiani et al. which supports both user-generated triggers in push or pull mode, and some automatic triggers based on contextual events [GPS12]. Meanwhile, *Panelrama* and *AdaM* are focused on automatic triggering based on the available devices [Par+18; YW14].

In our framework we support *user-initiated* distribution of UI components from any device currently running an application, thus we support both *push* and *pull* modalities. We also support the *automatic trigger* modality based on the proxemic relationships of the co-located devices captured by the IPS.

5. **Device Sharing between Multiple Users** – This dimension considers the cases in which there are various devices and some of them can be shared by multiple users. This can happen either because the same device is targeted by the UIs of their applications, e.g., when two users use the same large display as a target for a migration from their mobile devices, or different users access the same interface on the same device, e.g. when two or more users exploit the same wall-sized interactive screen by using their own devices. Sharing implies that the supporting environment is able to indicate what the shareable devices are, whether there is any conflict in their use, and provide some information regarding their state.

   The possible levels of sharing considered are: multiple users can move information on that device (*sharing by moving*), or can even interact with that device (*sharing by interacting*). In the case of our framework, both scenarios are possible. However, this is an area that still needs further refinement when it comes to better device discoverability, multi-application support and conflict resolution.

6. **Timing** – In this dimension the aspect considered is the time when a device change should occur in a cross-device configuration. An typical case is a migration that has to be carried out as soon as the migration trigger is sent from the source device (*immediate effect*) in order to achieve seamless continuity. Another case covers the possibility for the user to specify the time when to defer the change in the cross-device configuration (*deferred effect*). This could be useful when the target device is temporarily unavailable to the user, hence the effect will be delayed until a more appropriate time.

   The range of values for this dimension includes: *immediate*, *deferred*, and *mixed* (i.e., when both scenarios are possible). In the case of our framework, the timing is *mixed* because the distribution/migration can be done while from one device to other another one while both of them are running the application (i.e., it has an *immediate* effect), but the application can also be used in one device which is put away and at a later moment the previous state is migrated to a different device (i.e., it has a *deferred* effect).

7. **Interaction Modalities Involved** – This dimension analyses the modalities involved in the cross-device UI. There are three possible values: *Mono-modality* means that the devices involved in the cross-device access support the same single interaction modality, *trans-modality* means that different devices can support different modalities, but any device supports only one modality at a given time. *Multi-modality* occurs when the cross-device interface simultaneously supports two or more interaction modalities at least one of the devices involved.

   Most approaches have addressed mainly graphical interfaces with mouse, keyboard and touch inputs. However, there are some examples that break this norm. For instance, the *Proximity toolkit* support for marker-based motion capturing using specialized cameras and the *Kinect* motion controller for skeletal tracking can be used to extend the input modalities of an application [Mar+11]. *CDI* also uses the *Kinect* controller to recognize mid-air gestures for cross-device interaction [BPP17]. There have also been efforts that incorporate voice input [Ghi+14].

   In this case our proposed framework does not limit the options of the developers. They can have application that are *mono-modality*, *trans-modality* or *multi-modality*. In the case of our application prototypes they can be considered mostly *trans-modality* since they support traditional mouse and keyboard input on Personal Computers (PCs) and touch input on mobile devices. However, they can also switch these input types when running on a touch enabled PCs, or on tablets with a keyboard/touchpad cover, in which case they may be considered that *multi-modality* is available.

8. **UI Generation Phase** – This dimension specifies the phase when the UI is obtained so as to be rendered on the target devices. In the *design-time* case the UI is built in advance for each type of device, and then at run-time only the state has to be updated, in case of migration.

   The *run-time* case covers the situation where a run-time engine dynamically generates the user interface, according to the features of the target device. For instance, the research by Melchior et al. is a prime example of a model-based approach that generates the UI at *run-time* [MVV11]. An intermediate approach (*mixed* case) is also possible where the supporting engine dynamically generates the user interfaces beforehand for the different devices by exploiting some logical descriptions which have been created at design time.

   In the case of our framework we promote the use of web-based technologies and responsive design. The components of the UI are designed by having in mind that they may have to be displayed and used on different types of devices. However, the UI itself is rendered by the web browser itself in a way that matches the constraints of the device where the application is running by following the correct styling rules defined in Cascading Style Sheets (CSS). Therefore, we may consider that there is both *design-time* concerns and a *run-time* engine that renders the UI making it a *mixed* approach.

9. **UI Adaptation Aspects** – When changing the devices currently used, UI adaptation is usually required. The adaptation process can have an impact at various granularity levels: either the entire application is changed depending on the new context, or just some logical UI parts (presentation, navigation, content) or even single UI components are adapted. There is also the case that no adaptation is provided, often generating low usability results.

   There are three main approaches to adaptation identified within this dimension: *scaling*, in which the user interface is just linearly scaled according to the interaction resources of the available device, *transducing*, an approach preserving the initial structure while translating the elements into other formats, and compressing/converting images to match device characteristics (e.g., it is similar to responsive web design), and *transforming* which goes further to modify both contents and structures originally designed for desktop systems to make them suitable to display.

   Most tools do not go beyond a *transducing* approach [GPS12; Mel+09; Neb17; ND16]. In the case of our framework, there is no level adaptation that is imposed. It all depends on what the developers want to implement on an application by application basis. By using responsive web design within the applications and conditional *JavaScript* code execution it should be possible to emulate any of the three levels presented above.

   In the case of our prototype applications, the layout was adapted to different screen sizes using a responsive web design approach. For instance, a component with multiple columns would display a single column for easy scrolling on a smartphone. This roughly corresponds to the *transducing* approach.

10. **Architecture** – Two different strategies can be considered with regard to the architecture of a possible platform supporting a cross-device environment: *client/server*, in which there is an intelligent unit managing all requests and sending all data to target devices, thus controlling the user interface allocated across various devices; *peer-to-peer*, where the devices directly communicate and negotiate the distribution parameters (e.g., toolkit proposed by Melchior et al. [Mel+09]).

    A later revision of the framework introduces the *without fixed server* strategy [Pat20]. This corresponds to a situation in which there is still a *server* but it is not a fixed central one. Instead, it can migrate across the available devices when needed. The work presented by Frosini et. al and the *Connichiwa* framework follow this approach [FP14; Sch+15].

    In the case of our framework, it follows a *client/server* approach. There is a server-side component that is responsible for aggregating all the information about a running cross-device application and that continuously sends events to the connected instances on each of the participating devices. However, the engine that decides whether to show or hide certain components on a given device is actually run independently on each device based on the information they get from the centralized broker.

Paternò later presented a revised framework which includes eleven dimensions instead of ten [Pat20]. Most of the dimensions are the same but the following ones were removed (i.e., they do not have a direct match on the new framework): *Device Sharing between Multiple Users* and *Timing*. Meanwhile, the following dimensions were added:

- **User Device Access** – This dimension deals with the relationships between users and devices in a cross-device environment. These relationships can be considered *one-to-many* when the support is mainly for one user interacting with multiple devices. Another possibility is the *one-to-one* relationships which means that each user interacts with only one device, even if the effects of their interactions can modify the user interfaces of other devices (e.g, by distributing some component to them). There is also the *many-to-one* possibility when multiple users can interact with one device at the same time, (e.g. a public display that is shared by multiple users). Finally, there is the *many-to-many* possibility where multiple users can interact with public shared devices as well as with their personal device.

  An example of the *one-to-many* case is the *Improv* framework that allows end users to augment the user interface of an existing application on a primary device (e.g., a laptop or desktop PC) using the input elements of an additional mobile device (e.g., a smartphone or table) [CL17]. The applications of built with the help of the *Proximity toolkit* show some of the characteristics of *many-to-many* scenarios. In the case of our framework, we initially supported the *one-to-many* scenario, but we later added support for sharing application states between users thus opening up the *many-to-many* scenario.

- **Device Selection** – This dimension has to do with the way target devices are selected for distribution or migration operations. Some environments just provide the name of the available devices, but users may not understand what actual devices correspond to the identifiers listed.

  Other approaches provide the possibility of selecting devices from a *graphical representation* of the surrounding environment with the indication of where the devices are located in it, but this requires preparation of the graphical representation in advance [GP10].

  CDI supports the possibility of identifying target devices through gestures that can be immediate to perform [BPP17]. The possibility of connecting devices through physical touch has also been explored [Jok+15]. In fact, there are also commercial products that use Near-Field communication (NFC) to enable *Bluetooth* pairing via touch/very close proximity.

Panelrama addresses device selection by finding the best match between the features of the nearby devices and the types of information contained in the panels [YW14]. This is similar to the automatic approach employed by our framework.

Another possibility is to identify a set of devices through the role of their users [FP14]. For example, in a cross-device mobile city tour application, there is the tourist role and the guide role. The interactive elements shown to one role can be different from the ones shown to the other.

The prototypes built using the *Proximity toolkit* presents another approach based on the position and distance of the users and devices in the environment [Mar+11]. Our framework also incorporates proxemic information as part of its automatic distribution algorithm.

- **Multiple Roles** – This dimension addresses the need to consider user roles when distributing user interface elements. This is especially important in collaborative environments characterized by *many-to-many* device access. For instance, the framework presented by Frosini et al. can take into account both device types and users roles [FP14]. Meanwhile, *AdaM* allows developers to formalize the problem as the optimization of the assignment of user interface elements to devices taking into account aspects specific to user roles such as device access, user interface elements permission and privacy, as well as element importance and device characteristics [Par+18]. The optimization formulation and implementation use existing integer linear solvers which can search for solutions that maximize the objective function and satisfy the defined constraints.

2.1.2.2 4C Framework

Another interesting conceptual model is the *4C Framework* which identifies interaction principles for digital ecosystems [Sør+14]. Figure 2.1 presents the themes and principles covered by the framework.

The four themes and principles are the following:

- **Communality** – This theme covers situations of sequential interaction involving several users. It refers to cases where artifacts are shared between users, but with an emphasis on each user interacting with the artifact at a time. This could be in a public setting where communal computing has been used to describe computer resources made available in libraries. The concept can also be applied to public displays or tablets shared among family members. There are two principles in this category:
  - *Personalization* – It means that the relationship between users and artifacts is individual and tailored to each person. A common example of this is the use of accounts or profiles on network services like *Facebook*, where each user has access to something particular to their person.

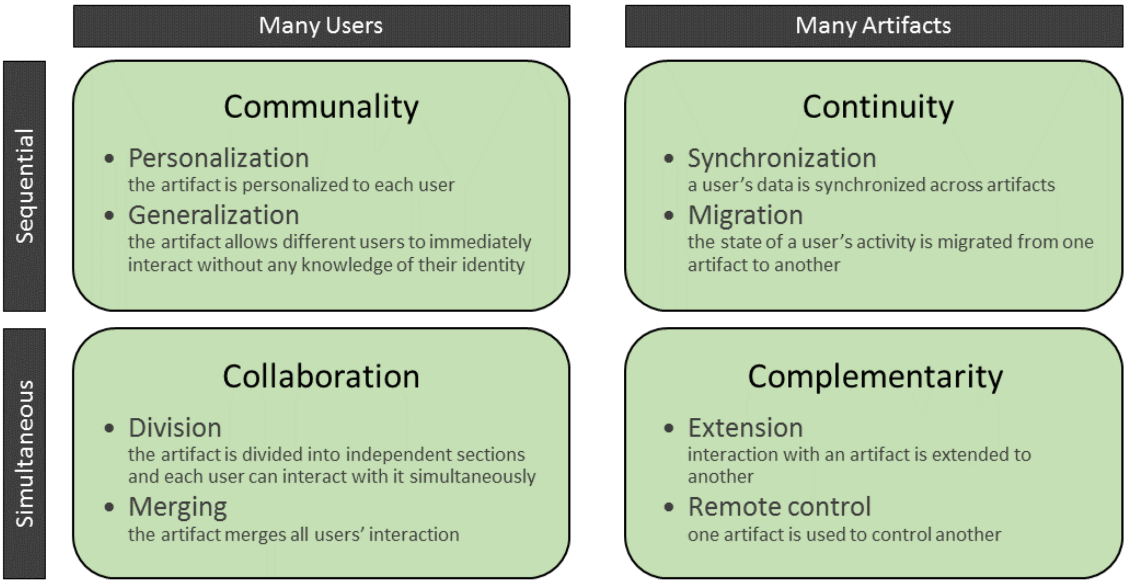

Many Users
Many Artifacts
Sequential
Simultaneous
Communality
• Personalization
the artifact is personalized to each user
• Generalization
the artifact allows different users to immediately interact without any knowledge of their identity
Continuity
• Synchronization
a user's data is synchronized across artifacts
• Migration
the state of a user's activity is migrated from one artifact to another
Collaboration
• Division
the artifact is divided into independent sections and each user can interact with it simultaneously
• Merging
the artifact merges all users' interaction
Complementarity
• Extension
interaction with an artifact is extended to another
• Remote control
one artifact is used to control another

Figure 2.1: The 4C framework of principles for interaction design in digital ecosystems [Sør+14].

  - *Generalization* – It means that the relationship between the artifact and the users is not a personalized one but the same as for everybody else. We use the principle to describe cases where an artifact can be used immediately by anyone without knowing who the user is. This could, for example, be in the case of a ticket machine at a train station, or the projector in a meeting room, which does not necessarily need to know who you are to provide its service or functionality.

  When it comes this theme our framework and prototypes follow a *personalization* approach since each user has to login into the application has access to its own data within an application. It is possible to share data with others, but it is an explicit action that each user must take to share with another specific user.

- **Collaboration** – The second theme covers situations of simultaneous interaction by many users, which we refer to broadly as cases of collaboration. Collaborative use of digital artifacts has long been a topic within the area of Computer-Supported Cooperative Work (CSCW), and like this research field the term collaboration is used in a broader sense than describing just ways of working together and coordinating activities, to include all kinds of social computing situations for recreational and social activities. Broadly speaking, simultaneous collaborative interaction in a digital ecosystem is for the purpose of doing, or engaging in, a shared activity or task involving joint interaction with one or more shared digital artifacts. The two principles that are part of this theme are:

  - *Division* – It means that the interaction with an artifact is split between users and provides them with individual parallel points of attention. The most common example of this principle is spatial partition of graphical user interfaces, such as split screen views in multi-player video games, or large displays with separate workspaces, where people can interact independently through different views. Division can also be done by other means, such as separated audio channels, where different users hear different sound.
  - *Merging* –It means that several users' simultaneous interaction with an artifact is done *over the top of each other* through one shared representation. This could, for example, be in the case of a multi-user board game on a shared tablet where several users' interactions are merged visually into one, or the case of a large mixing desk where different people can jointly and simultaneously manipulate the sound of different instruments through merged physical controls.

  Regarding this theme, our framework supports the *merging* principle since all users manipulate one shared application state at a given time and the multiple displays are not split so that multiple users can use them for different purposes. However, this is something that we intend to explore in future work when it comes to shared public displays that can in principle be split into smaller virtual devices to support concurrent users or groups of users.

- **Continuity** – This theme addresses situations of sequential interaction involving several artifacts. It refers to cases of continuity, where an interaction starts on one artifact and then continues on another. This enables people to use several artifacts and re-access content on a different device. Such continuous interactions can be facilitated by keeping data consistent across artifacts, or by allowing activities started on one artifact to be continued on another one, exactly where it was left off. This theme also has two possible principles:

  - *Synchronization* – It means that data in a digital ecosystem is kept consistent across all devices. When an artifact synchronizes with an ecosystem, content and its organization is replicated to this artifact, and when changes are made on one artifact this is applied to all other artifacts as well. Well-known commercial examples of the synchronization principle in digital ecosystems are cloud-based storage services like *Dropbox* or *Google Drive* where one's files are automatically replicated, or made accessible, across devices. Other examples include email and calendar services that facilitate continuity in the interaction from one artifact to another by synchronizing information content.
  - *Migration* – It refers to allowing users to switch between artifacts by transferring the state of their activity or interaction from one artifact to another, either partially or completely. For instance, *Amazon Kindle* allows people to continue reading a book on one device from where they left off on another one. Another

example of migration is *Google Cast* where one might browse media on an *Android* device and pass on its playback to a large display or sound system that has a *Chromecast* device plugged in.

In the case of our framework we support *migration* of the state between devices but this is achieved through the *synchronization* of the data that encapsulates the state information of the application's UI.

- **Complementary** – The last theme is about simultaneous interaction with multiple artifacts and the effect created when using several digital artifacts together as one. This refers to the situation where interaction with one artifact adds to the interaction with another artifact, and these jointly make up a larger whole. There are two principles in this category:

  - *Extension* – It describes the case where one digital artifact directly adds to another one. This could simply be the use of several smartphones and tablets to create a larger display area, or the use of a companion app to provide supplementary functionality for another device.
  - *Remote Control* – Complementarity is achieved by one digital artifact simply controlling another, as is well known from traditional TV or sound system remotes. While perhaps seemingly trivial, this principle of interaction in a digital ecosystem is in fact very common, and many companion apps provide exactly this functionality for media players or home automation systems.

  Regarding *complementary*, our framework does not directly impose an *extension* or *remote control* approach. It will depend upon how each application is designed. Developers are free to follow only one of the principles or even a little bit of both.

2.1.3 Applications Supporting Migratory User Interfaces

We have already briefly introduced the concept of migratory user interfaces an of the types of cross-device applications. Their main motivation comes from the realization that users must restart what they were doing every time they change their device. The objective of migratory user interfaces is to move between a *source* and a *target* device while preserving the state from the *source* device.

The state of the user interface that should be preserved after migration involves the results of all the possible user actions, including the scrolling position and the element in focus, and potentially the entire application environment, which, in the case of applications running on a web browser, means cookies, sessions, history, bookmarks, and *JavaScript*.

For instance, *Myngle* eases the transition of web navigation between desktop computers and mobile devices through browser extensions and native applications [Soh+11]. It shows visits to previously accessed pages through an integrated history functionality that records interactions across multiple devices. It then allows users to search and filter through the history according to high-level categories.

In fact, the need for a better integration and transition between mobile and desktop browsing has been recognized by browser vendors. Nowadays, browsers such as *Mozilla Firefox*, *Google Chrome* and *Microsoft Edge* have desktop and mobile versions of their browsers which are capable of syncing bookmarks, history, list of open tabs, passwords, among other items. They also often allow sending an open page from the desktop to the mobile browser.

Deep Shot is a framework and run-time which follows a different approach to support task migration [CL11]. A picture of the current task on the desktop computer is taken from a smartphone. From that picture *Deep Shot* uses computer vision to detect the application that is running and some information about its state. It is then able to send this information to the smartphone as an Uniform Resource Identifier (URI) that encodes the application state. The smartphone can then use this information to resume the task.

In another approach, users can access any web application with the support of a proxy server which injects to support migration [GPS13; GPS12]. Through a migration client users can determine the target device for migration. When the migration is triggered, the state of the current page is sent to the migration server which prepares a version of the page for the target device so that the users can resume their activities. This solution also supports partial migration of only a subset of its elements. It has also been further expanded to allow automatic or assisted migration based on context changes (e.g., proximity between devices based on *Bluetooth* signal strength) [GPA13].

2.1.4 Applications Supporting Cross-Device User Interfaces

As mentioned in 2.1.1, *distributed user interfaces* support application access through multiple devices at the same time. These type of interfaces can help users to collaborate, perform a task on the most suitable device available, or use one device in coordination with others. However, for an user interface to be to really considered *cross-device*, there must be a way for the various parts of the distributed user interfaces to be kept in sync. Redirection and composition of input events across devices should also be possible.

CDI is an example of a framework which promotes the development of applications with this type of user interface [BPP17]. It is a *JavaScript* library that provides an Application Programming Interface (API) to support gestures and cross-device interactions. It provides the possibility of defining combined events that associate events occurred in different devices and determine the transfer of state information across the involved devices.

Liquid Software is an approach in which applications and data are supposed to flow seamlessly across devices, allowing users to continuously switch from one device to the next without having to worry about how to transfer applications, settings or associated data between them has also been referred to as *Liquid Software* [HMP96]. The term was coined by Hartman in the 1990s [HMP96; Har+99b], but it has resurfaced in more recent research about ubiquitous computing [Gal+16; MSP15; TM15; TMS14]. Moreover, this concept ties well with cross-device applications and cross-device user interfaces.

Web-based technologies are often employed to build this type of applications since they are naturally capable of running across a wide variety of devices. Besides, the challenge of creating software that runs fluidly on different screen sizes and input modalities is partially addressed by responsive Web design [MK11]. However, there are still many challenges that remain unanswered, such as, code mobility, state synchronization, UI adaptation and shared data access [MSP15].

The *Liquid.js* framework is a prime example of previous work enabling the development of cross-device applications that operate on a shared decentralized state using web-based technologies (e.g., *WebSockets*, *WebRTC* and *Web Components*) [GP16]. It aims to support sequential screening (applications that run on different devices at different times), simultaneous screening (applications running on different devices of the same user at the same time) and collaboration (multiple users collaborate using the same application on all of their devices at the same time).

It is also conceivable to build cross-device ecosystems using native tools at the cost of having to re-implement certain components across multiple platforms. Apple's *Handoff* feature enables sequential screening between an *iPhone* and *Mac* [App]. It is the only widely available solution that embraces part of our vision. However, on top of applications needing to be specifically written to take advantage of *Handoff*, a specific version needs to be installed on each type of device.

2.1.4.1 Distribution of User Interface Components

The distribution of user interface components across devices can be specified at various times [Pat20]:

- *Design time* – The specification of how the various parts should be allocated to the involved devices is pre-determined when the application is designed.

- *Run-time* – The distribution is obtained obtained through dynamic customization tools, which enable end-users to determine distribution configurations not considered at design time.

- *Mixed/Hybrid* — The definition of the distribution is made partially through the definition of event handlers at *design time* that are triggered by specific events at *run-time*.

We have found previous work specifically focused on the distribution of content across multiple displays. For instance, *Dygimes* is an environment for model-based UI development [Con+03; VC04]. The models are used to develop specifications at design time, and the executable UIs are obtained at runtime by exploiting such models.

Another example is *Panelrama* [YW14]. It introduces a lightweight specification that allows programmers to provide additional semantics for HTML elements. It also categorizes device features which allows an interface optimizer to allocate the various elements to those devices that best correspond to the desired features. *AdaM* follows a similar approach in which users or developers need to provide additional semantics for each interface component to the constraint solver [Par+18]. However, while *Panelrama* considers only single users interacting with multiple devices, *AdaM* targets collaborative multi-user applications. *XDBrowser* is another cross-device example, segmenting Web pages and distributing the fragments across devices [Neb17].

Manca et al. proposed another model-based solution [MP16]. It extends the *MARIA* to allow the specification of UI distribution at multiple granularity levels: user interface, single presentations, composition of elements, single elements or parts of single elements [PSS09]. It takes into consideration the *CARE* properties (*complementarity*, *assignment*, *redundancy*, and *equivalence*) which can be changed by users to affect the distribution of elements to better suit their needs [Cou+95].

There are also solutions that focus on particular domain such as *Vistribute* [Hor+19]. This solution provides a data analysis framework that is capable of dynamically adapting itself to a multitude of devices based on in-depth information about the views and the data they visualize. *Vistribute* is based on *Webstrates* which enables sharing media across multiple devices by synchronizing the browser's underlying *Document Object Model* (DOM) tree [Klo+15].

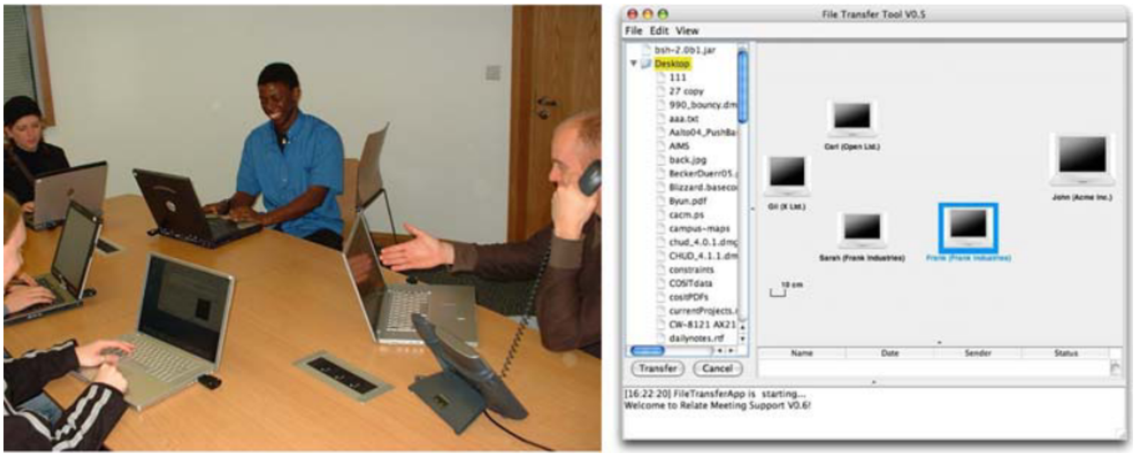
In the case of the *YanuX Framework*, the automatic distribution follows a *design time* approach that adapts itself to the available devices in the surrounding environment based on pre-determined restrictions and preferences for each of the UI components of an application. However, it also follow a *run-time* approach by allowing users to manually override the automatic distribution to fit their needs.

2.1.4.2 Representation and Selection of Devices

Another important issue that should be tackled is the representation of available devices in a cross-device environment in way that users can understand and which allows them to perform tasks that involve those multiple devices. One of the most interesting models that deals with this issue is called *RELATE* [Gel+08]. It is designed to support spontaneous interaction of mobile users within their immediate surrounding environment. It aims to help users understand what devices and services are present, and to support association of the user's mobile devices with any of the devices in a seamless manner. The devices involved can be situated devices such as printers and public displays, as well as co-located mobile devices.

The interaction model requires the spatial relationship of a user's devices with other co-located devices to be known. Therefore, the authors have developed a relative positioning system based on bi-directional ultrasonic ranging, synchronized over a dedicated

Radio Frequency (RF) channel. They developed a proof of concept application which provides users in a meeting with a spatial interface to easily interact with other participants (Figure 2.2).



[16:22:20] FileTransferApp is starting...
Welcome to Relate Meeting Support V0.6!

Figure 2.2: Meeting situation on the left, and the corresponding spatial user interface on the right [Gel+08].

The application supports matching the names of participants with their relative position around the table, chatting with spatially selected participants, transferring files by spatial reference, and connecting to a large shared display. Users can identify the desired target device on their screen by mapping the real world arrangement of devices to the corresponding layout of icons in the interface. The map view also supports direct manipulation, such as the selection of one or more of the devices, or *drag and dropping* files onto device icons.

They developed another application to support the discovery of co-located devices and services. Discovered devices are represented by *RELATE Gateways* which are arranged around the edge of the screen using a compass metaphor (see Figure 2.3). The position in the interface indicates the direction of the device providing the service. The gateways can be used to access the services provided by the corresponding device.

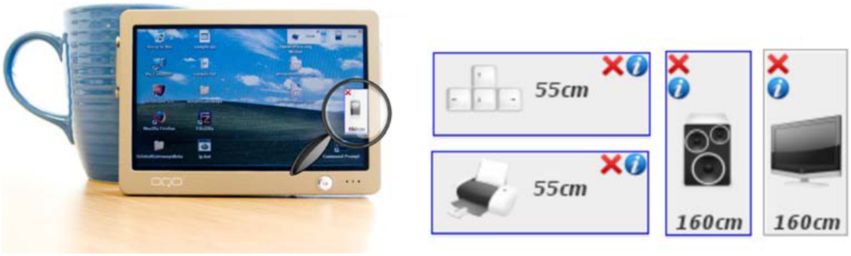


Figure 2.3: Gateways arranged at the periphery of the user interface of a mobile device (left) and examples of gateways to a variety of services (right) [Gel+08].

The authors also performed an experiment to determine whether the use of spatial references for the selection of co-located devices was advantageous. They compared how users reacted to a list of network discovered devices and services in three different situations (see Figure 2.4).

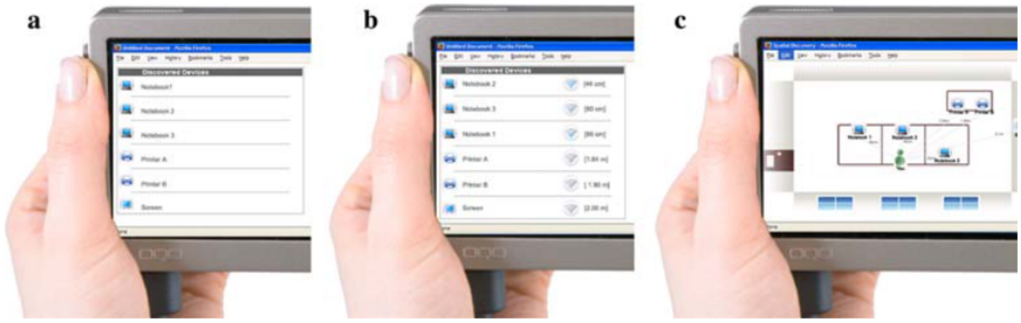


Figure 2.4: The three interfaces implemented on an handheld to study the use of spatial references: **a**) Alphabetically sorted list; **b**) List sorted by device distance; **c**) Iconic map view [Gel+08].

The results did not show a significance preference for the iconic map view, and were consistently lower for the list sorted by device distance. It was probably easier to search the alphabetically sorted list because the device names were clearly visible to the users. Nevertheless, two thirds of the participants selected the iconic map view as their most preferred interface for the selection task.

Gradual Engagement also addresses the issue of visualizing available devices but it goes further than that. It consists of a design pattern which considers the fine-grained relationships between multiple devices and allows the seamless transition from awareness to information transfer [Mar+12]. Engagement increases continuously across three stages as users move and orient their personal devices towards other surrounding devices, as depicted in Figure 2.5.

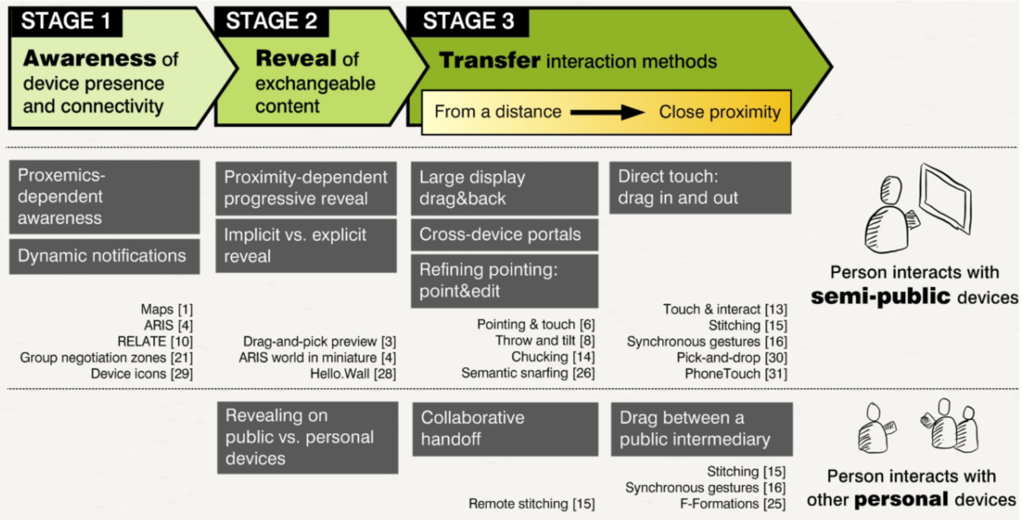

STAGE 1
Awareness of device presence and connectivity
STAGE 2
Reveal of exchangeable content
STAGE 3
Transfer interaction methods
From a distance
Close proximity
Proxemics-dependent awareness
Dynamic notifications
Proximity-dependent progressive reveal
Implicit vs. explicit reveal
Large display drag&back
Cross-device portals
Refining pointing: point&edit
Direct touch: drag in and out
Person interacts with semi-public devices
Maps [1]
ARIS [4]
RELATE [10]
Group negotiation zones [21]
Device icons [29]
Drag-and-pick preview [3]
ARIS world in miniature [4]
Hello.Wall [28]
Pointing & touch [6]
Throw and tilt [8]
Chucking [14]
Semantic snarfing [26]
Touch & interact [13]
Stitching [15]
Synchronous gestures [16]
Pick-and-drop [30]
PhoneTouch [31]
Revealing on public vs. personal devices
Collaborative handoff
Drag between a public intermediary
Remote stitching [15]
Stitching [15]
Synchronous gestures [16]
F-Formations [25]
Person interacts with other personal devices

Figure 2.5: The gradual engagement design pattern's stages[Mar+12].

The different stages of engagement in a cross-device environment, and the multiple interaction techniques that are proposed by the authors, can serve as a very important starting point to understand what kind of abstractions should be available for information transfer between devices within the scope of our work.

Just like *Gradual Engadgement*, *GroupTogether* is an example of how developing applications that spawn across multiple devices overlaps with the area of proxemics interaction being specifically concerned with the proper selection and interaction between devices. It builds upon the concepts of *f-formations* and *micro-mobility* [MHG12]. This research area and concepts are presented on section 2.2.

The system enables fluid co-located collaboration by leveraging the relationships between users and their devices. It uses both on-device sensors (accelerometers and radio transceivers) and extrinsic sensors in the environment (*Kinect* camera and fixed-location radios) to achieve this goal.

The project is focused on the implementation of cross-device interaction methods, and in particular on how to decide when devices should be federated (by sensing *f-formations*) and how that information should be shared (by sensing *micro-mobility*). The explored interaction modes include:

- **Tilt-to-Preview** - It provides a way to transiently share selected digital content across devices by tilting devices toward others. The receiving user can grab a copy of the transiently shared information.

- **Face-to-Mirror** - When someone holds their tablet vertically, the interactive canvas is mirrored at full-screen scale to the display of all other tablets of the group.

- **Portal** - When tilting a tablet towards the device of any other group member, a tinted edge appears along the shared screen edge of the two slates. By dragging an item through this edge and releasing the touch, the item is transferred permanently to the other device.

- **Cross-device Pinch-to-Zoom** - It allows viewing content across multiple tablet devices using a pinch-to-zoom gesture. A person can use a two finger pinch gesture to enlarge any content on the screen, but when the sender enlarges the zoomed content beyond the visible area of the slate's display the remaining content expands onto the surrounding tablets.

The interactions modes that we just described can be extended to work with more than two users. For example, a person can share content with a large group by tilting their tablet towards the center of the formation, rather than just tilting towards a single person. Users can also share content with the digital whiteboard in a manner analogous to sharing content to slates held by other participants.

Ghiani et al. also explore how to target devices in a cross-device environment [GP10]. They focused on representations of the surroundings and the available devices rather than determining their relative positions. This means that positions of the devices in the environment was defined at design time with only the position of the user changing at run-time. They considered two alternatives: a *map-centered* approach in which the map of the environment has a fixed position and orientation, and the representation of a user changes as they move; and *user-centred* where the user icon is in the center and never changes its position or orientation but the map around the icon translates and rotates to reflect their movement.

Spatial information can be useful to more easily select the target devices in cross-device interaction, rather than using lists with device identifies that are often difficult to keep track of given the devices currently available in the environment. Therefore, in order to assess the importance of spatial representations, Rädle compared cross-device interaction techniques with and without spatial information [Räd+15].

The technique without spatial information was similar to the one proposed in *Conductor* [HW14]. It uses menus with color-coded device names to select the tablets to share information with or to chain tasks across them. The other two tested techniques were made spatially aware thanks to the existing *HuddleLamp* system [Räd+14]. One was based on a color-coded radar view of the devices present in the environment. The other technique showed colored bubbles around the edges of the screen that followed the positions of the devices in the surroundings. Their findings suggest that spatial information improves usability. They can decrease mental demand, effort, and frustration during mobile cross-device interactions but must be implemented with care.

Another study also compared three distinct methods to transfer visual objects between a smartphone and a tablet [Jok+15]. The results indicate that there strongly depend on the task and the context of use, making the design of a single optimal cross-display object movement method a challenging task. It all depends on the number of objects to move, the number and characteristics of the devices involved, the social situation and its implications in terms of privacy and security.

2.1.4.3 Decentralization

The attempts at implementing cross-device user interface solutions are often dependent on the access to a centralized remote server. However, the access to a server may not always be possible (e.g., if the server is accessible through the internet and an application must work offline). Communication over the internet also introduces latency, which is especially undesirable on interactive applications, and additional security concerns.

Frosini et al. propose a framework which allows to target devices for the distribution of UI elements in various ways: specific device identifiers, device groups identifier, specific users, and user groups associated with specific roles [FP14]. It does not use a remote server to support UI distribution. Instead, one of the devices holds the distribution engine component, which receives the distribution requests and maintains the current distribution state. This engine component can migrate from one device to another if required.

Therefore, it can work offline through an ad hoc network without internet connectivity.

Connichiwa is another solution which does not require a remote server [Sch+15]. It runs local web applications and a native helper application automatically runs a web server on one of the devices. Other devices can then access the web server through a shared network. This eliminates the need for a remote server and keeps latency to a minimum.

There are also solutions which try to go one step further and do not have a server at all. Not even a local server running on one of the devices and serving the others. Instead they rely on the nodes being capable of coordinating with each other by following a *peer-to-peer* approach [Mel+09]. This improves flexibility and fault tolerance at the cost of additional complexity with the UI state distributed across multiple devices.

2.1.4.4 Integration of Smartwatches and Wearables

One of the most recent device categories that has seen a widespread adoption is that of smartwatches. Therefore, it is interesting to consider the research that has been done in the area of cross-device user interfaces that involves these extremely personal devices with small screens and various sensors. For instance, *WatchConnect* supports the development of cross-device interaction techniques and applications based on smartwatches [HM15]. It provides developers with an extendable hardware platform that emulates a smartwatch, a UI framework that integrates with an existing UI builder, and a rich set of input and output events using a range of built-in sensor mappings.

Similarly, *Duet* explores the design space concerning interactions between a smartphone and a smartwatch [Che+14]. It takes into consideration their spatial configurations in order to coordinate touch-based interactions with the two types of devices. As a result, the smartwatch enhances various smartphone-based interactive tasks. For example, the interface between the phone and the watch can be divided using the smartphone to show a canvas while using the smartwatch to host a palette. It supports cross-device gestures such as allowing users to mute the two devices at once by performing a *pinch-to-close*. The system can also differentiate which part of the hand is touching the smartphone display by using the smartwatch sensors.

Finally, we would like to mention *Weave* which focus on wearables in general [CL15]. For instance, besides smartwatches it also includes support for *Google Glass* smartglasses. The framework supports abstractions for selecting target devices, performing output actions on selected devices, and handling input events on one device. Each device require a native application to be installed that provides runtime support for the cross-device interactions.

2.1.5 Context-aware Adaptation

In the scope of human-computer interaction and ubiquitous computing, context can be defined as "... any information that can be used to characterize the situation of an entity.

An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [Dey01]. Therefore, the contextual information in an environment with multiple devices is an important aspect to consider in order to better adapt the user interface to each situation. As part of the contextual information we can identify the following dimensions [Pat20]:

- *User* – It involves aspects related to the task, mental state, physical activities, preferences or knowledge of the user(s).

- *Environment* – It relates to aspects concerning the actual physical environment and its attributes such as noise, light, and temperature.

- *Devices* – The available devices and their features such as screen size, connectivity, multimedia support, and orientation.

- *Social aspects* – The various possible relations (e.g., friendships and social groups), which may be relevant to deciding what to show in the user interface.

Any element in these dimensions can have an impact on how to change the design in order to improve the user experience. User interfaces are composed of various aspects that can adapt dynamically, including but possibly not limited to:

- *Presentation* – It consists of how the elements are arranged and the associated attributes. This may achieved by allowing users to personalize the arrangement of their user interfaces for specific devices [NSN13].

- *Dynamic behavior* – It indicates how it is possible to navigate across the available elements. For instance, there are techniques such as direct guidance, or dynamic hiding/showing links, or links annotations [Bru01].

- *Content* – It provides the information communicated. Adaptation may be achieved through page summarization with the objective of automatically reducing content when needed [Buy+02].

A possible way to structure adaptation based on context information is through the use of *events*, *conditions* and *actions*. An *event* is something that happens at a given time which occurs by interacting with the application or in the surrounding context of use. A *condition* is a specific constraint on the state of some contextual aspect that must be satisfied. The action defines the desired effect, e.g., the activation of some functionality or a desired change in the user interface. *Events* and conditions can then be combined with logical or temporal operators to trigger certain *actions* [CR14; CR15].

In the context of cross-device applications the *action* may be a set of changes in the user interface which can involve the entire UI, some UI parts, the relations between some UI elements (e.g., changing their order), single UI elements, or only attributes of single UI elements.

There have been proposals that apply this type of approach to cross-device user applications by implementing user interface changes and distribution across different devices allowing users to create rules such as: "When I am close to the TV duplicate the smartphone user interface on it" [Ghi+17]. In fact, this is similar to the approach promoted by our framework. However, the rules are implicitly defined as roughly as "When I am close to a device that meets the requirements of one or more UI components, as defined by the developer, show them on that device".

An aspect that is particularly relevant in cross-device user interfaces is the spatial relations amongst the involved entities. This leads us to the concept os proxemic interactions which is based on a set of spatial relationships between people, objects and devices [Gre+11; Mar+11]. This aspect was embraced as a core part of our framework. Therefore, there is a section that is dedicated to its exploration (see 2.2).

2.1.6 Authoring Cross-device User Interfaces

Regarding the creation of cross-device applications and user interfaces there are multiple approaches that have been followed [Pat20]. A possible approach is to perform *platform-specific authoring*, i.e., develop variants of the UI for each of the supported platforms. It has the advantage of allowing better control over final results, but the work has to be repeated for each of the supported platforms which may be an unbearable effort for many projects and development teams.

Another possibility is to follow a *model-based authoring* approach [Fon+10]. They use user interface conceptual descriptions in order to avoid dealing with many low-level implementation details for the various devices addressed, and use interface generators to render adapted user interfaces. These approaches enable designers and developers to concentrate on the main conceptual aspects and avoid using a multitude of coding languages by linking semantic information and implementation elements.

Supple is an example of a model-based approach [GWW10]. It is an intelligent tool able to take as input the indication of the supported tasks, the constraints specific to the platform considered, a relevant example scenario, and a cost function reflecting user preferences, and finds the design able to optimise the cost function still satisfying the constraints associated with the considered platform.

MARIAE is another example which supports editing and analysis at the various possible abstraction levels, and then user interface generation for different modalities and implementation languages [PSS11]. There are also approaches that try to extract a logic model from the implementation on a specific target platform which is then used as the basis to generate user interfaces for other platforms [KK11; Mes+08].

One popular approach to addressing the variety of available devices is responsive design, which is particularly oriented to Web applications [MK11]. Through the use of CSS media queries, it is possible to detect several characteristics of devices, such as screen width, height, orientation, aspect ratio, colour, resolution, type of pointing device, etc

[Riv+12]. There also other CSS features such as *Flexbox* and *Grid* that can be used to build responsive layouts [Tab+18; Tab+20].

Responsive web design is the approach adopted by our framework since we encourage the use of web-based technologies. Each of the UI components that a developer wishes to have distributed across multiple devices should be designed by having in mind the kind of devices where it may be deployed. However, by ignoring certain optional components of our framework it is also possible to follow a platform-specific approach without using any web-based technologies for the UI.

The final approach is *distributed authoring*. It allows developers and designers to e to edit and show the user interfaces distributed across the various possible platforms at the same time. *MultiMasher* applies a mash-up approach to cross-device environments [Hus+14]. It supports the development of cross-device web applications based on the reuse of existing web sites created for single device usage. However, it is not possible to create new components for the cross-device user interfaces.

Another example is the *XDStudio* which supports two modes when designing distributed user interfaces [Neb+14b]. The simulated mode enables designing for a cross-device environment while working on a single one device which simulates the other devices. The on-device mode distributes the design process across multiple devices at the same time, since design and development are conducted directly on the target devices. The same authors have also explored session management in cross-device interfaces including the support for testing and debugging this type of applications [Neb+15].

2.2 Proxemic Interactions

The concept of proxemic interactions evolved from the broader anthropological concept of proxemics introduced in 1966 by Edward Hall [Hal90]. It has however also been applied to other fields such as HCI and Human-robot Interaction (HRI). This concept has also been used context of ubiquitous computing and cross-device applications to enrich their user experience.

We considered the inclusion of proxemic relationships as part of our framework from the very beginning in order to allow developers to build smarter applications that are more enticing to use. As such, this section presents the concept of proxemics in the context of ubiquitous computing along with previous work and existing challenges.

2.2.1 Theoretical Background

Proxemics identifies the culturally dependent ways in which people use interpersonal distance posture and orientation, to understand and mediate their social interactions with each other [Hal90]. When considering the cultural dependency, Hall argues that "Space perception is not only a matter of what can be perceived but what can be screened out. People brought up in different cultures learn as children, without ever knowing that they have done so, to screen out one type of information while paying close attention to another." [Hal90].

The theory has many aspects, but the definition of four proxemic zones characterizing how people interpret interpersonal distance is possibly the most relevant to HCI (see Figure 2.6): *intimate* (0 to 0.45 m), *personal* (0.45 to 1.2 m), *social* (1.2 to 3.6 m), *public* (3.6 to 7.6 m and beyond). In general, closer distances should lead to increasing interpersonal engagement and intimacy. People adjust these distances to match their social activities and to raise defense mechanisms when others intrude into these zones.

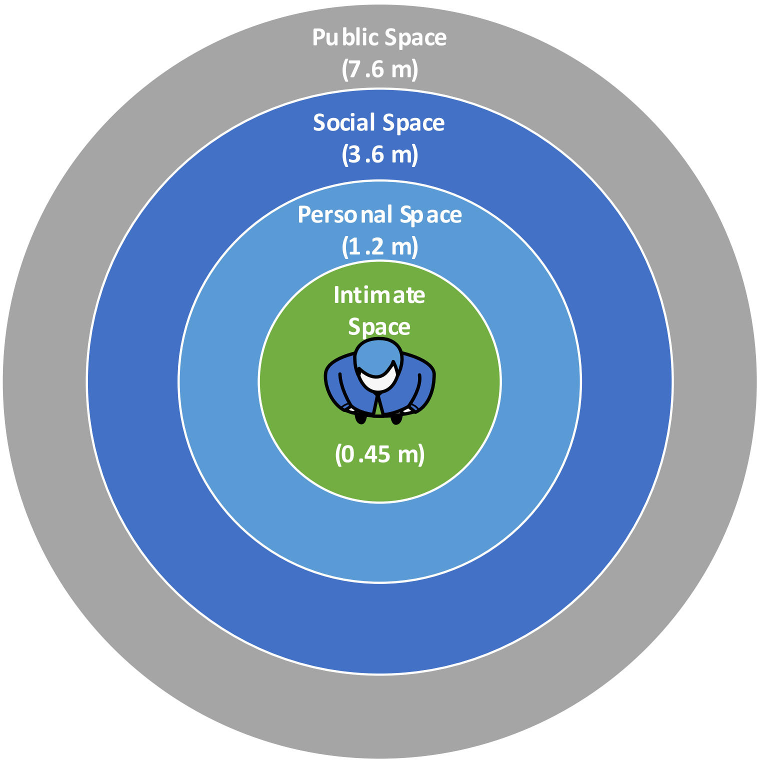


Figure 2.6: The four proxemic zones as described by Edward Hall [Hal90]

Hall also described how features within the space affect people's interactions. Those include *fixed* features of places where people tend to organize certain kinds of social activities (e.g., entrance to a room) and *semi-fixed* features which can bring people together or move them apart (e.g., the arrangement of chairs). He noticed that such features might affect space usage, where the layouts can be *sociofugal* (separating people) or *sociopetal* (bringing people together). These features may also interfere with how people behave and perceive their surrounding space while interacting with ubiquitous applications.

Nowadays, the vast majority of devices is unaware of the presence of other devices, people or other non-digital objects and features in their surroundings. However, such awareness could affect the devices' intended use. Just as people expect increasing engagements and intimacy as they approach others, they should also expect increasing connectivity and interaction possibilities as they bring their devices in close proximity to one another.

Greenberg et al. applied proxemics to HCI and ubiquitous computing by defining the concept of *proxemic interactions* [BMG10; Gre+11]. According to them, *proxemic interactions* can be seen as not only describing spacial relationships between people, but also as "devices with fine-grained knowledge of nearby people and other devices - their position, identity, movement, and orientation - and how such knowledge can be exploited to design interaction techniques" [BMG10].

In the context of ubiquitous computing, only the proximity between entities such as digital devices, physical objects and people need to be dealt with. Therefore, some aspects of the original theory that concern only people, such as social and cultural elements, can be ignored. This leaves us with the following proxemic dimensions defined by Greenberg et al. [Gre+11]:

- *Distance* – It can be a continuous measure (e.g., a value between 0 and 3 m) or it may be discrete (e.g., one of the proxemic zones).
- *Orientation* – It can be continuous (e.g., the pitch, roll or yaw angle of one object relative to another) or discrete (e.g., facing forward, facing sideways or facing away from other object).
- *Identity* – It can be a detailed measure with the exact identity, a less detailed measure, such as an entity's type, or a minimal measure that distinguishes an entity from others.
- *Movement* – It captures the distance and orientation of an entity over time.
- *Location* – It describes an entity's physical context (e.g., a particular room and its characteristics).

There are authors that have expanded upon these five proxemic dimensions by introducing concepts like *F-formations* and *micro-mobility* [MHG12]. *F-formations* concern the distance and relative body orientation among multiple users, which indicate when and how people position themselves as a group during face-to-face interaction.

Each *F-formation* consists of two or more persons engaged in a joint activity. Their bodies define three regions (see Figure 2.7): the *O-space* reserved for the main activity pursued by the group, the *P-space* which determines group membership, and the surrounding *R-space* which buffers the group from the outside world.

Micro-mobility is concerned with the fine-grained orientation and repositioning of objects so that they may be fully, partially viewed, or concealed from others. It is postulated that moving an artifact closer to others, and orienting or subtly tilting it towards or away from them, affords powerful and nuanced ways for individuals to share information, to fluidly manage the focus of conversation and to make their intentions clear to others.

By combining and integrating these concepts one can arguably build applications with a better user experience. These are also perfectly mapped to the vision of cross-device

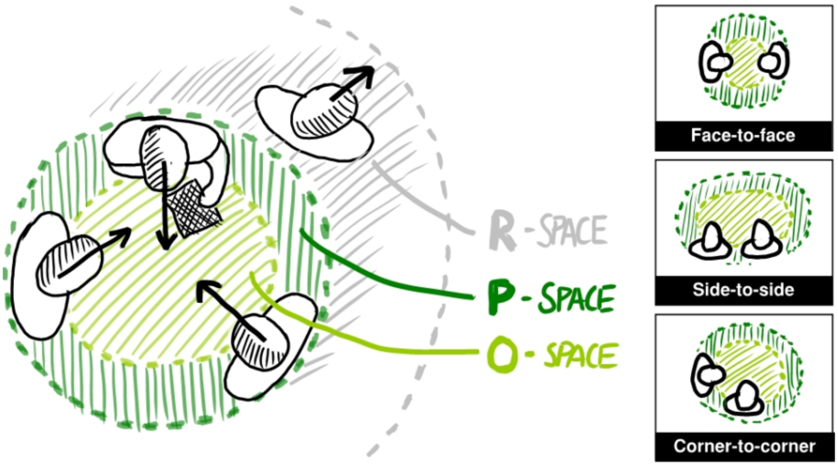


Figure 2.7: Example of an F-formation[MHG12].

applications presented in Chapter 1. Therefore, this research area is an important part of our research.

2.2.2 Existing Work in Human-Computer Interaction and Ubiquitous Computing

Although the study of proxemics was first presented in the realm of social environments, the emergence of ubiquitous computing disclosed the potential of proxemics in the scope of HCI. By letting people expect a higher connection and interaction possibilities when they bring their devices near each other, just like when people approach each other expect a higher level of interpersonal social interaction, proxemics application in HCI offers the growth of context awareness, an essential feature in ubiquitous settings [Gre+11].

Some of the earlier work based on proxemics comes from Greenberg and Kuzuoka in the late 1990s when they were exploring the concept of digital surrogates as a away to express activity awareness between distant people [GK99]. Their *Active Hydra Unit* establishes a permanent audio and video connection between distant colleagues. Proximity to the unit is used to naturally adjust the balance between awareness and privacy.

The audio and video fidelity is controlled as a function of each person's distance to the device, thus mimicking the proxemic zones. Both people can see and hear each other clearly when they are both close to the units. As they move away, audio is disabled and if they move further away the video quality is degraded so that it is only possible to know that someone is present but it is impossible to grasp finer details. *MirrorSpace* follows a similar approach by creating a video communication mirror which sharpens or blurs the image depending on the distance from the display [REH04].

ViconFace [DG10] is another project by Greenberg and his team in which a smiley face reacts to multiple proxemic events captured by their own *Proximity Toolkit* [Mar11; Mar+11]. This toolkit is built on complex and expensive motion capture technology and it has served as the basis for many more application prototypes.

The *Proxemic Media Player* introduced a home media player application running on a large display, which people could interact with through *implicit interactions* or *explicit interactions* [BMG10; Gre+11; MG12].

In terms of *implicit interactions*, the *Proximity Media Player* adapts the interface shown depending on the user's proxemic relationships. For instance, as the user enters the room it shows an overview of what media is available for playing, when he/she gets closer to the screen the overview shows smaller thumbnails and up close it gives detailed information about each video allowing him/her to select what to play. When the user sits on the couch, the selected video switches to full-screen. The media player automatically pauses the video when the user faces away from the screen, answers the phone, or starts talking to someone else.

If someone else walks into the room the player shows the title of the current video, so that the newcomer knows what is playing. As she moves closer more detailed information will be shown beside the video. Finally, if everyone leaves the room the video is stopped and the display is turned off. Regarding *explicit interactions*, users could select a new video by pointing out an object at the screen and media can be seamlessly transferred by approaching a portable media device to the large display.

The *Proxemic Peddler* is a digital peddler application which incorporates continuous proxemic measures, including distance and orientation, while also taking into consideration attentional states (e.g., digression and loss of interest), and the passersby interaction history, with the goal of improving the chances of leading someone to buy the advertised products [WBG12]. As part of this effort, the authors tracked users with *Vicon* motion capture cameras to capture the proxemic relationships, used animations to get and regain the users' attention, and promoted interaction by displaying increasingly personalized information as users approached.

The *Proxemic Pong* is a re-imagined version of the *Pong* video game which reacts to the distance, orientation, motion and identity of the users [Gre+11]. It dynamically creates a paddle for each person that approaches the display. Players take turns at trying to stop the ball from getting off the screen through the bottom of the playing field. Players that get too far from the screen are removed from the game.

Mueller et al. focused on using proxemics to support the design of engaging games and digital play experiences [Mue+14]. By analyzing games such as *Musical Embrace*, which engages players (preferably strangers) to hug a shared pillow-like controller in order to advance in the game, the authors explore how this type of game can create experiences based on proxemic zones.

AirPlayer offers a multi-room music system built on top of *Apple*'s *AirPlay* [Sør+13]. It enables music playback to follow the user around the house. It can also display and change the music playing in the room the user is in. The authors suggest that proxemic interactions have a great potential for hiding parts of the interaction with a complex system, but it is challenging to simplify background interactions so that details about the interaction are still apparent.

XDKinect uses *Microsoft*'s *Kinect* motion controller for multi-modal interaction involving multiple users and devices [Neb+14c]. It provides an APIs for inter-client communication, supporting cross-device sessions involving multiple distributed clients and their proxemic interactions. For example, it was used to adapt the display and interaction modality (personal or ambient interaction mode) according to the user distance to a wall-sized display.

Public displays can also leverage proxemics to react to the users' interest by analyzing their position, orientation and social context [Dos+14; Sch+13]. In fact, the same display can even be used as a public and personal display depending on the context. Vogel et al. developed an interaction model that seamlessly moves users from distant implicit interaction with public information to up-close explicit interaction with personal information across four continuous phases [VB04] (see Figure 2.8): *Ambient Display*, *Implicit Interaction*, *Subtle Interaction*, and *Personal Interaction*.

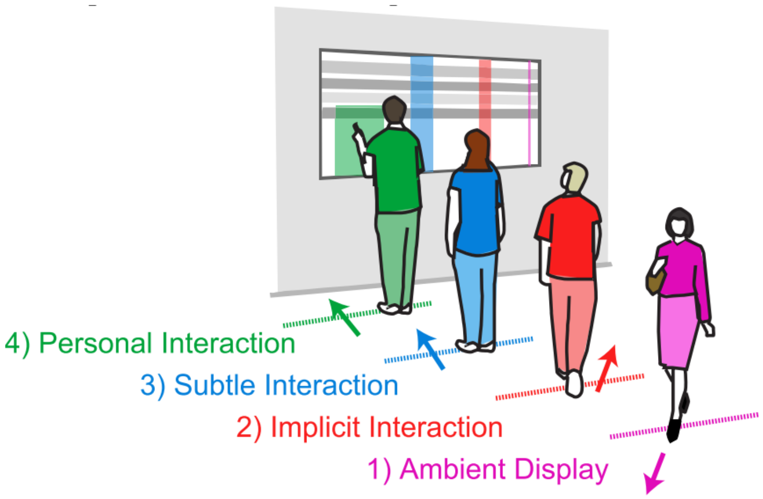


Figure 2.8: Vogel's four interaction phases[VB04].

Ju et al. followed a similar approach by defining an implicit interaction framework and applying it to an interactive white board application which reacts to the users' proximity according to four different zones inspired by Hall's proxemic theory [JLK08; JLK07].

The concept of interaction zones is also used in the standard approach of the *iBeacon* protocol based on BLE. It classifies a user's proximity to a specific Point of Interest (PoI) into four proximity zones: *immediate* (0 to 1 m), *near* (1 to 3 m), *far* (more than 3 m) and, *unknown* (device not ranged) [Zaf+17].

The interaction with mobile projectors [Löc11] and the interaction with interactive walls and multi-touch tables alongside displays of various sizes to form a multi-display environment [Räd13] are also worth mentioning as possible areas of research where proxemic dimensions need to be considered.

Even the World Wide Web can be adapted to react to the proxemic relationship between its users and the devices that are used to access it. The *proxemic web* is an extension to the concepts of responsive web design which goes beyond screen size and form factor [SVK14]. However, this is an idea still in its infancy with many issues that still need to be addressed, such as what should the distance thresholds be and what kind of input mechanisms are supported.

2.2.3 Challenges Surrounding Proxemic Interactions

Even though there is a considerable amount of work done in the area of proxemic interactions, there are still challenges that need to be overcome [Gre+11]. For instance, many of the interaction techniques are based on the interpretation of how certain entities are related. However, it is hard to always choose the right interpretations that meet user expectations due to the implicit nature of those interactions. A trade-off between implicit and explicit interaction must be reached.

The changes in the proxemic relationships, and its impact in the application and user interface, is also a concern. For instance, the reaction to position and distance variations can occur as continuous movements (system interactions follow the changes in the distance) or discrete proxemic zones (space divided into regions, affecting the interaction by entering/leaving them) [BMG10]. However, it is not always obvious what is the best approach. In fact, it will probably depend on the situation and task at hand.

The information that applications receive about the environment may be imprecise, noisy or inaccurate which may lead to a poor user experience. For instance, there may be jitter along the borders of different interaction zones thus creating a frustrating interaction experience for users. Therefore, designing robust proxemic-aware applications can be very challenging. This is especially true in the context of this dissertation, since we intended to use readily available and inexpensive technology whenever possible which means that accuracy and precision are diminished when compared to more sophisticated approaches.

The usage of proxemics can also potentially be subverted to create bad user experiences, invade the user's privacy, or to violate other social mores. Designers should always consider the negative and positive effects of these technologies. For instance, users risk inadvertently exposing their activities when using public facing displays. Therefore, Brudy et al. proposed a system in which users are alerted about the presence of *shoulder surfers* behind them, while also providing the possibility to hide content that is not directly shielded by the user's body [Bru+14].

A series of other problems and insights on what should be avoided to build pleasant proxemic applications are identified in [Bor+14] and [Gre+14]. In general, the applications should be well received by users and perceived as being safe to use. They should not be too intrusive and an unwanted presence in their lives. They must respect people's conception of physical space and make it easy for users to opt-in and out of them.

Moreover, as a specific type of ubiquitous computing applications, proxemic applications face many of the same challenges [MG12]. These include finding ways to reveal interaction possibilities, to direct actions at the system, to establish connection between devices, to provide feedback to the users, to avoid and correct mistakes, and to deal with security and privacy issues.

However, proxemics also present an opportunity to do things in ways that were not previously possible. For instance, in order to tackle the issue of how one should enable connections between devices, proximity measures as distance and orientation can be exploited [MG12]. Proxemics can also be used to drive interaction with visualizations by keeping track of the spatial relationship between an user and a large wall sized display. As it was studied by Jakobsen et al. this can provide a useful and pleasant experience in certain situations [Jak+13]. Nevertheless, proxemic interaction may sometimes lack the level of control required by the users.

2.3 Indoor Positioning Systems

With the emergent growth of ubiquitous environments and the need to power them with seamless interactions, knowing the location of a device at all times is increasingly becoming a requirement. In this context, the awareness of devices in indoor environments has been a target of multiple studies and research leading to indoor positioning systems. Moreover, the continuous tracking of people and devices is essential in the design of proxemic interactions in a cross-device computing environment. Therefore, an IPS was needed in order to allow an interactive system to be extended beyond the confines of a single device. However, it is generally hard to determine an entity's position while indoors.

While Global Positioning System (GPS) has become ubiquitous, it does not work well indoors because there is no line of sight to the satellites and signal attenuation [Bre+17; Küp05; Mau12]. Moreover, despite extensive research throughout the years, there is a lack of a universally accepted solution for indoor positioning. Therefore, a solution which takes into account the trade-offs between accuracy, precision, refresh rate, cost and ease of deployment is needed to realize our vision. We were especially interested in using Commercial off-the-shelf (COTS) devices and in avoiding major infrastructure changes. As such, this section presents concepts related to indoor positioning and an overview of the state of the art which covers these requirements.

2.3.1 Types of Indoor Positioning Systems

Indoor positioning system are often split by authors into categories. For instance, Location-based Systems (LBSs) which provide services to users based on their location can be distinguished by the type of location estimated. A system can focus on inferring the user's exact location, known as micro-location, or provide an estimation of the user's proximity to a specific PoI, known as Proximity-based Services (PBS).

The *Bluetooth Special Interest Group* divides them into *proximity solutions* and *positioning solutions* [Blu19]. A *proximity* solution is one that leverages a technology to determine the position of two devices according to each other, in particular, if one device is within range of another device. On the other hand, positioning solutions use the technology/technologies to assess devices' real physical location.

Brena et al. divided such systems into *active solutions*, where a device generates the signal instead of receiving it and *passive solutions* where a device receives the signal instead of generating it [Bre+17].

Though the labeling of indoor positioning systems can diverge, depending on the author's research context, it is accepted that an indoor positioning system always provides information about the place where a user or object is situated in an indoor environment.

2.3.2 Measurement Principles in Indoor Positioning

According to the Cambridge Dictionary, a measurement is "a value, discovered by measuring, that corresponds to the size, shape, quality, etc. of something." [Cam21]. Brena et al. define the action of measuring as an evidence stage, where "devices involved measure characteristics of a signal." [Bre+17].

In indoor positioning, there are many different types of measures that can be taken to be used by the various positioning techniques. The following are possibly the most used as part of indoor positioning systems:

- **Time of Arrival (TOA)** – The absolute time taken by the signal to go from the transmitter to the receiver.

- **Time Difference of Arrival (TDOA)** – The time difference of arrival from synchronized transmitters.

- **Angle of Arrival (AOA)** – The angle at which a signal is received in a reference device.

- **Received Signal Strength Indicator (RSSI)**: – It represents a signal's intensity as measured by the receiver. The Received Signal Strength Indicator (RSSI) is a term used to measure the relative quality of a received signal to a client device, but has no predetermined unit. Each chipset manufacturer can define their own arbitrary scale for RSSI values. Therefore, more often than not, the RSSI values are expressed in *decibel-milliwatt* (dBm) which represents a logarithmic relative power level expressed in *decibels* (dB) with reference to one milliwatt (mW).

2.3.3 Positioning Methods

An IPS should continuously determine the position of objects or people located in a physical space by relying on one or more positioning methods [GLN09]. This subsection presents some of the most commonly used methods for positioning in indoor environments.

2.3.3.1 Proximity Sensing

The position of an object is determined based on the closeness to a reference point. It is assumed that if a receiver is able to detect a signal from a transmitter, then it must be within the coverage range of the transmitter. The position of the receiver is then considered to be the position of the transmitter, or the average of the positions of multiple transmitters within range. This is a very straightforward method but it tends to have a low accuracy. An estimate of how close the receiver is to the transmitter may also be provided based on the received signal strength.

2.3.3.2 Lateration

Circular lateration uses measurements from at least three reference points for 2D positioning and four for 3D. In 2D, the positions of the receiver and of the i^{th} transmitter are denoted as (x, y) and (X_i, Y_i), respectively. The distance between the receiver and the transmitters is determined by Equation 2.1, where n is the number of transmitters. This results in a circle of radius r_i around each transmitter where the receiver may reside (see Figure 2.9). The intersection of the multiple circles is the position of the receiver.

$$r_i = \sqrt{(X_i - x)^2 + (Y_i - y)^2}, i = 1, \ldots, n \tag{2.1}$$

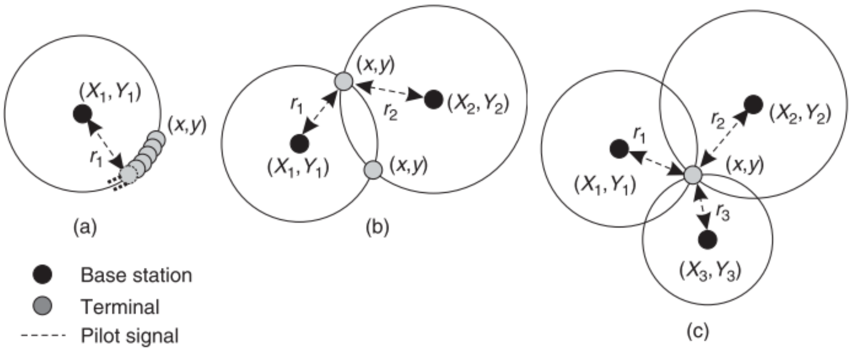


Figure 2.9: *Circular lateration* in 2D [Küp05]

In *hyperbolic lateration* the measurements consist of distance differences [AI11]. This means that only then set of base stations needs to be time synchronized. The range to the first base station is given by r_1, and the range to the second one by r_2. The difference

$(r_2 - r_1)$ limits the target's position to a hyperbola (see Figure 2.10(a)). and the intersection between two hyperbolas delivers the target's position (see Figure 2.10(b)). Three hyperboloids would be needed in 3D.

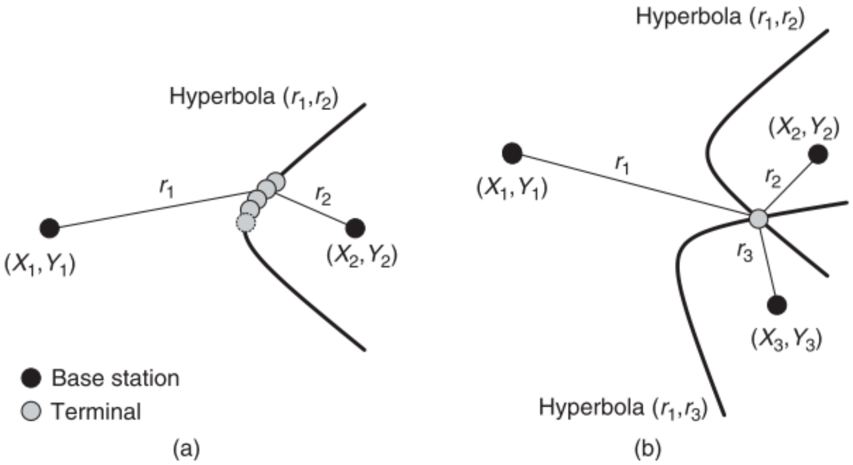


Figure 2.10: *Hyperbolic lateration* in 2D [Küp05]

Measured distances often deviates from the actual value due to measurement errors. Therefore, the system of equations may not have a solution. To solve this we can start from a rough position estimate, linearize the system of equations and apply the *linear least squares* method to approximate a solution [Küp05; KPV11; Wer14].

2.3.3.3 Angulation

In *angulation*, angles are measured between known base stations and the target device [Küp05; Wer14]. The signal's angle can be measured at the base station using an antenna array, which restricts the target's position to a line that intersecting the target's and the base station's position. Angle measurements to two base stations are needed to obtain a position in 2D (see Figure 2.11).

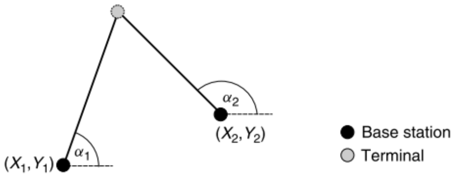


Figure 2.11: Angulation in 2D [Küp05]

2.3.3.4 Dead Reckoning

Dead reckoning extrapolates the current position from the previous position, direction of motion and the velocity of the target, or the traveled distance. Since a previous position must be known, *dead reckoning* must always be used together with another positioning method [AI11]. The direction, distance, and speed of motion may be deduced from previous positions, or using additional sensors, e.g., accelerometers and gyroscopes. *Dead reckoning* has the advantage of allowing a device to operate autonomously. Once the starting position is obtained, no external information is required. However, measurement errors will accumulate and lead to large positioning errors.

2.3.3.5 Fingerprinting

Fingerprinting relies on collecting signals at reference points. The mobile terminal estimates its location by matching measured signals with the recorded radio map. This approach is common with solutions based on Wireless Local Area Networks (WLANs). The access points' RSSIs are typically collected at reference points spaced around 1 m apart [Var+07].

There are two main types of matching algorithms. In deterministic algorithms, an access point's RSSI at a point is characterized by a scalar value and non-probabilistic approaches are used to estimate the user location [Rox+07]. Algorithms in this category include: *nearest neighbor* [BP00]; *k nearest neighbors in signal space* [BP00; Hos+07; Rox+07] and *smallest polygon* [Rox+07].

Probabilistic algorithms incorporate movement history or map information when characterizing the RSSI. They usually employ *Bayesian inference* to determine the target's location [AI11]. The location from the radio map that is chosen will be the one with the highest probabilities based on stored RSSI distributions at each location in the radio map [Hae+04; YAU03; YA04].

Fingerprinting-based techniques are more resilient to non-line-of-sight propagation than the positioning techniques previously discussed [KPV11]. It is possible to achieve errors below 2 m and latency can be kept relatively low [AI11]. RSSI values can also be easily obtained without the need of hardware or software modifications.

Scalability to a large numbers of users is easy, but it is hard to scale to large areas due to the time needed to collect samples. However, signal propagation models can be used to reduce the number of required reference locations at the cost of reduced positioning performance [BP00; Hos+07]. There are also proposals on how to deal with device heterogeneity, since different devices can provide inconsistent readings at the same location [Ras+12].

2.3.3.6 Computer Vision

Computer vision can track objects by using one or several cameras. There are two possible approaches [Wer14]: a camera is given to the mobile device and the location is extracted from its point of view (*inside-out tracking*); or cameras are part of the infrastructure and information is extracted from the location of a person or object in the camera stream (*outside-in tracking*).

For mobile camera systems, features extracted from the camera's pictures are compared with a database of features referenced to locations. Some approaches put markers into the environment, others use naturally occurring markings. In infrastructure-based systems, an empty scene is recorded as the ground truth and the images captured are used to extract features to identify objects of interest. The position of the objects is then tracked and translated into real-world coordinates [Hu+04; PVB04; WHT03]. Tracking people has also been studied in the areas of surveillance, security, authentication, and context awareness [Ham+05; Hu+04].

Under ideal circumstances, *computer vision*-based techniques provide positioning accuracies in the range of a few centimeters. However, results can be highly influenced by environmental conditions such as lighting and the presence of occluding objects [KPV11]. Moreover, *computer vision*-based methods do not implicitly provide the identity of a tracked target. Therefore, additional steps must be performed, e.g., facial recognition.

2.3.4 Technologies Used in Indoor Positioning Systems

The research on indoor positioning systems brought forth the application of a comprehensive set of technologies that is used to find the position of.

In the survey made by Mautz or, more recently, by Brena et al., the authors listed the following technologies as the most commonly used in indoor location-based solutions [Bre+17; Mau12]:

- **Radio Frequency Technologies** – Technologies that exploit the frequency of radio signals. It includes *Bluetooth*, *ZigBee*, *Wi-Fi*, Radio-frequency identification (RFID), and Ultra Wide Band (UWB).

- **Sound-based Technologies** – Technologies that use sound waves as *Audible Sound* and *Ultrasound*. There are also *passive* approaches which use sound without embedded information, e.g., takes already available sounds in the environment as characteristic of a given place and uses a database of known places.

- **Optical Technologies** – Technologies whose core implies optical signals as *Infrared* and *Visible Light*. There are also *passive* approaches which such as the position of known light sources like lamps to find a location.

- **Computer Vision** – An IPS that uses the information collected by cameras and image processing techniques for identifying and tracking objects.

- **Magnetic Field** – Both natural Earth's magnetic field, along with its irregularities, and artificially produced magnetic fields.

- **Inertial Technology** – These technologies are often employed to implement the *dead reckoning* positioning method (see 2.3.3.4) by using *accelerometers* and *gyroscopes*.

The list above is in no way extensive. There are other possibilities that are less common. There are even systems that combine multiple technologies and positioning methods to achieve better results.

We now present *Wi-Fi* and *Bluetooth* technologies, with a special focus on BLE, since they were the ones adopted in the IPS presented in Chapter 4.

2.3.4.1 Wi-Fi

Wi-Fi is a radio frequency-based technology for WLANs that follows the *IEEE 802.11* standard. This wireless technology, named by the organization *Wi-Fi Alliance*, was first released in 2000 through a set of products following the *IEEE 802.11b* standard. This particular standard and set of protocols are considered one of the most crucial developments in the information age. *Wi-Fi*'s primary goal is to provide networking communication through a wireless signal, in which devices communicate with an Access Point (AP), typically a router, that completes the connection to other devices on Local Area Networks (LANs) and Wide Area Networks (WANs) such as the Internet.

Wi-Fi transmits information on the 2.4 GHz and 5 GHz *ISM* unlicensed radio bands, typically using 20 MHz wide channels. Although 2.4 GHz still remains the most used frequency for *Wi-Fi* applications, it can potentially conflict with other wireless technologies like *Bluetooth*, boosting the popularity of 5 GHz and, more recently, 6 GHz.

Wi-Fi's potential to achieve high precision, low-power consumption, and low-cost solutions, aligned with its transmission range of 100 m, make this technology suitable for application in fields other than network communication, including indoor positioning [Xia+17]. *Wi-Fi* has established a solid reputation, rising to become a technology with particular dominance in indoor positioning research. In such cases, the *Wi-Fi* access points are usually exploited by taking advantage of metrics such as the quality of signals received from them (e.g., the RSSI values).

Furthermore, *Wi-Fi*'s application for indoor positioning can take advantage of two different signal metrics [Xia+17]:

- **Received Signal Strength Indicator (RSSI)** – The strength of the signal received from an AP is used to determine the distance that separates the AP and the receiver. Multiple measurements can be combined to determine a concrete position.

- **Time and Space Attributes of Received Signal (TSARS)** – Time and space attributes of a received signal from an AP are used in the positioning computation such as Time of Arrival (TOA) and Time Difference of Arrival (TDOA).

Due to the ease of availability, *Wi-Fi* solutions based on RSSI values are some of the most common indoor positioning solutions. Time and Space Attributes of Received Signal (TSARS) approaches often require modified hardware and/or software stack making them less practical. However, there have efforts to standardize these approaches as part of the *IEEE 802.11-2016* revision of the *IEEE 802.11* standard. These new features allow the measurement of the TOA between a device and an AP. However, both of them must support this new feature which is still not widely deployed.

Despite the growth of other radio frequency technologies as *Bluetooth* and *ZigBee*, *Wi-Fi* is still the most represented technology in indoor positioning research. Table 2.1 presents an overview of the benefits and issues related to the application of this technology to indoor positioning settings.

Table 2.1: Wi-Fi Overview

Benefits [Bre+17; FH15; Xia+17]	**Shortcomings** [FH15; Mar+20; Xia+17]
Good precision	Not optimal for movement tracking (limited update rate)
Ubiquitous technology	Privacy may be an issue
Low-power consumption	Suffers from multipath fading
Low cost	New Wi-Fi power-saving techniques can reduce accuracy

Furthermore, *Wi-Fi* technology has a crucial distinctive characteristic when comparing with other technologies: it has been intensely scrutinized in a vast range of studies, allowing for a better consensus on the benefits and challenges associated with it.

2.3.4.2 Bluetooth and Bluetooth Low Energy

Bluetooth is a radio frequency wireless standard for *Wireless Personal Area Networks* used in data exchanging between devices over short-distances settings. Firstly developed in 1994, it is currently managed by the *Bluetooth Special Interest Group* [McD05]. This short-range communication technology has been continuously improved to provide high security and low-cost solutions. With the *Bluetooth* 4.0 specification, the appearance of *Bluetooth Low Energy* (BLE) gave developers the ability to start leveraging *Bluetooth* to create robust, low-cost real-time location systems [Hey13].

BLE technology operates in the same spectrum range as classic *Bluetooth* technology, but uses a different set of channels. Instead of the classic *Bluetooth* seventy-nine 1 MHz channels, BLE uses forty 2 MHz channels. Classic *Bluetooth* and BLE share the same 2.4 GHz frequency as other wireless technologies like *Wi-Fi*, which may cause interference problems.

The low-cost features associated with BLE derive from the reduced battery consumption, which is obtained by the communication of short duration messages [Hey13]. These messages can be of two different types: *data messages* or *advertisement messages*. The latter enables the broadcast of messages used for device discovery, which is required to form any communication. Moreover, BLE advertising occurs indefinitely on channels 37, 38, and 39, at 2402 MHz, 2426 MHz, and 2480 MHz, respectively. This is done to reduce the interference with other signals, such as *Wi-Fi* [FH14].

The presence of *Bluetooth* technology in indoor location solutions has been increasing at a steady pace since the release of BLE. The *Bluetooth Special Interest Group* foresees over 400 million products related to this type of solution by 2022 [Blu19]. There has also been the introduction of low power BLE transmitters powered by small batteries which are usually called beacons. These beacons commonly implement a protocol such *Apple*'s *iBeacon* on top of BLE [App18].

BLE initial IPS solutions were based on the Proximity Profile (PXP) and on the aforementioned *iBeacon* protocol by using the RSSI from beacons to assess the distance separating them from users [Blu15; FH15; Zaf+17]. Moreover, the *Bluetooth Special Interest Group* asserts that the standard approach leveraging *Bluetooth* uses "Bluetooth Low Energy radio to determine if two Bluetooth devices are in range of each other and, in many cases, use received signal strength (RSSI) measurements to estimate the distance between the two devices." [Blu19]. According to *Bluetooth Special Interest Group*, there are two main types of approaches [Blu19]:

- **Positioning Solutions** – Beacons are deployed throughout the environment and continuously transmit *Bluetooth* signals. The users are then able to listen for these locator beacons via an application. Based on which beacons the application can detect, alongside the RSSI transmitted from each beacon and their known locations, the application can use a positioning technique as *trilateration* to calculate the user's current position.

- **Proximity Solutions** – The users have low-power *Bluetooth* transmitters called tags attached to them, which are programmed to transmit a signal periodically. The beacons connect back to a centralized server, reporting which tags they can detect and the corresponding RSSI. Based on a positioning technique as *trilateration*, the server can then estimate the position of the tags and thus the users' position.

Table 2.2 provides an overview of the benefits and shortcomings associated with the employment of this technology to indoor positioning solutions.

Table 2.2: Bluetooth Overview

Benefits [Bre+17; FH15; Jia+14; Tec20]	**Shortcomings** [FH15; Mar+20]
Low power consumption	Multipath effect on the three advertising channels
Ubiquitous technology	Long window scanning size
Low cost	Uneven channel gain
Good precision in short range	Presence of fast fade

2.3.5 Performance Metrics

Most indoor positioning solutions have focused on accuracy as the prime performance metric. However, accuracy alone is not enough to measure the performance of indoor positioning systems. According to Mautz, "a crucial element for any initiative to design an indoor positioning system is a thorough study of the user requirements and specific application descriptions in order to justify the research and development in this field" [Mau12]. Additionally, the author suggested the weighting of 16 user requirements as performance metrics (see Figure 2.12).

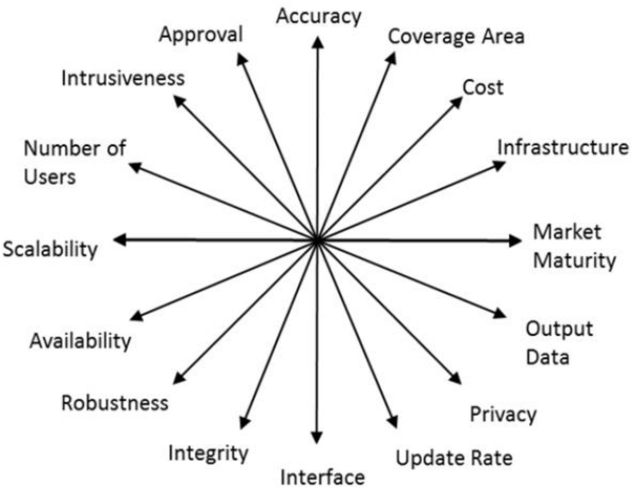


Figure 2.12: Performance metrics suggested by Mautz [Mau12]

Some of the most relevant user metrics that can form the basis for the requirements of an indoor positioning system are:

- **Accuracy** – Often considered the primary and most impacting measure of an indoor positioning system. Generally, it computes the difference between the estimated position and the actual one. According to Mautz, accuracy should be considered as the degree of conformance of an estimated position at a given time to the real value, expressed at the 95% confidence level (P_{95}). Other important metrics include the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

- **Update Rate** – The update rate is the frequency with which the positions are calculated on the device or at an external processing facility. The update rate can be *on request*, i.e., triggered by the user or by a remote device, *periodic*, i.e., a regular update at a specific interval (e.g. 10 times per second = 10 Hz), or *on event*, i.e., measurement update initiated by the local device when a specific event occurs (e.g. when a temperature sensor exceeds a critical threshold).

- **Latency** – The system latency describes the delay with which the requested information is available to the user. In the case of IPSs and navigation applications the latency should usually be kept to a minimum. The positions should be available in *real-time*. However, there are situation in which positioning systems may have more relaxed latency requirements.

- **Coverage Area** – The area in which the system can locate a user or device. IPSs usually locate a device in an area of a few meters meters, a room, a floor or a building.

- **Availability** – It represents the percentage of time during which the indoor positioning system is available for use while maintaining its requirements.

- **Robustness** – It defines the extent to which the system can maintain its performance requirements within an acceptable margin under unexpected/undesirable circumstances (e.g., damage, hardware failures or software errors).

- **Scalability** – Judges whether the system can adapt to a increasingly larger environments and number of users while maintaining a certain degree of performance.

- **Cost** – This is an important requirement if the system intends to have a widespread adoption. The cost attached to an indoor positioning system should be kept as low as possible. The use of COTS components is preferred over more complex dedicated hardware.

2.3.6 Main Challenges

The development of indoor positioning systems is affected by a combination of challenges that can severely reduce its performance [Bre+17; Mau12; Xia+17].The following aspects represent some of the most common obstacles that this type of systems must face:

- Indoor environments are full of objects made out of a wide range of materials, shapes, and sizes that can attenuate the signal strength in unexpected ways.

- The constant change of the environment can influence solutions that need a mapping of the environment as the method of *fingerprinting*.

- The existence of signals mixed with their reflection, known as *multipath*, can cause difficulties in recognizing the characteristics of such signals.

- The lack of line of sight can cause several fluctuations in the values of RSSI from transmitters.

- The range of different devices and transmitters (e.g., beacons and access points) with variable transmitting and receiving characteristics make it difficult to create models that generalize well across so much potential heterogeneity.

2.3.7 Existing solutions

This section presents some examples of indoor positioning solutions categorized according to their primary positioning method and underlying technology.

2.3.7.1 Fingerprinting-based systems

RADAR is a widely known example of RSSI *fingerprinting* based on *Wi-Fi* [BP00]. The system can determine the position of mobile devices with a median accuracy of 2 to 3 m. Measurements were collected approximately every square meter during training and the deterministic *k-nearest neighbors* algorithm was used for matching.

RSSI variation due to the orientation of the user's body, the number of locations where data is collected, and the number of RSSI samples collected at each location have all been identified as important as aspects [BP00]. The authors also tested generating the fingerprint database from a propagation model to reduce the number of measurements needed. Unsurprisingly, the performance gets degraded to a median accuracy of 4.3 m.

HORUS uses probabilistic analysis and mitigates channel variation using correlation, continuous space estimation and small-scale compensation, with an accuracy of 1 m for 80% of the cases [You04; YA04; YA05]. It also uses location-clustering to lower its computational requirements.

COMPASS applies probabilistic methods but focus on the influence of the user's body orientation [Kin+06]. It samples the signal strength for selected orientations at each reference point. Only histograms with an orientation similar to that of the user are selected for position estimation. This approach is able to reach an accuracy of 1.65 m.

Saha et al. compared the matching performance of a nearest neighbor algorithm, a neural network, and a probabilistic approach [Sah+03]. The neural network outperformed the others with an error of less than 1.5 m 83% of the time with three access points, and 1 m in 72% of the cases with eight access points.

Evennou et al. use particle filters to obtain smoother trajectories as the target moves in a building [EMN05]. The technique constrains the positions on a *Voronoi diagram* of the building to avoid trajectories through obstacles and to ensure more consistency on sudden velocity variations.

Chai et al. reduce the number of reference locations and samples captured at each location [CY05]. They then use interpolation to estimate the missing RSSI values. Accuracy only decreases between 6% and 16% when the number of collected samples is reduced to one-third.

Instead of relying solely on a set of qualified users that carefully collect the data at specific reference points and have the duty to maintain the quality of the created radio map, there have also been proposals that follow a crowd-sourcing approach to ease the burden of creating and maintaining radio maps. In this cases, the responsibility of collecting data belongs to a broader set of contributors, in some cases, to the entire community. However, the quality of these radio maps tends to be poorer.

The *Where@UM* is a *fingerprinting*-based system based on crowd-sourcing that tackles the liabilities associated with this approach, namely the reliability in users' honesty and commitment to contribute to a good quality radio map [MM15]. Users may not always be well intentioned, making the radio map a target of attacks if not adequately protected.

The authors state that just a few erroneous contributions can disrupt the entire system. *Where@UM* system deals with these problems by building a collaborative radio map, taking into account the variable quality of user contributions, including its credibility and user reputation.

Chen et al. propose RSSI sample collection assisted by strategically placed RFID readers [Che+05]. They also deal with environmental changes by training multiple radio maps according to the conditions reported by sensors. Results demonstrate an error reduction of 2.6 m with respect to traditional *fingerprinting* systems.

Kaemarungsi et al. present a model that provides guidelines on how to design and deploy *Wi-Fi*-based positioning solutions given a set of variables, including the number of access points and the standard deviation of RSSI [KK04]. Concerning the effect of the number of access points on the performance measures, the authors assess that a system does not need more than five access points with complete coverage to achieve an adequate performance level.

In this line of thought, the *MMLOC* system provides a solution capable of reducing the required number of well-deployed access points [YRN19]. The key concept is to generate more fingerprints with fewer access points, designated as smart access points. These are equipped with two-mode antennas, and each smart AP is modified to be able to provide two RSSI fingerprints by shifting between modes. Additionally, the study provides a clustering-based localization strategy to inhibit the need for complex synchronization when recognizing the access points modes. This work shows that the number of required access points can be reduced by half while maintaining accuracy.

Xia et al provide an in-depth look at the application of *Wi-Fi*-based *fingerprinting* for indoor positioning [Xia+17]. The authors provide an analysis of the benefits of *fingerprinting* (low-cost and high precision) and disadvantages (heavy labor cost to maintain the radio map) and the influence factors in *fingerprinting* such as signal attenuation from people's presence in the environment. Furthermore, the authors introduce the main methods of collecting data: *empirical modeling*, which is the standard approach of measuring the signal strength received from different access points at every reference point, and *accurate modeling*, which only needs the signal strength of some crucial positions to generate the entire radio map.

Besides *Wi-Fi*, *Bluetooth* can also be used for *fingerprinting*. This is typically done by using BLE beacons placed in the environment instead of relying on pre-existing *Wi-Fi* AP. Other than that, the principle is the same with BLE having some advantages over *Wi-Fi*, For instance, the longer duration between *Wi-Fi* scans (a single scan can take several seconds) restricts the update rate on users' positions, whereas BLE can achieve much faster rates [FH15].

Faragher et al. tackles the shortcomings of *Wi-Fi fingerprinting* by providing a new analysis and proven arguments of *Bluetooth*-based *fingerprinting* potential [FH14; FH15]. In their studies, the authors research the impact of BLE devices in advertising mode on fingerprint-based indoor positioning schemes and also propose a comparison between

Wi-Fi and BLE *fingerprinting*.

Furthermore, the studies show how to mitigate fast fading in BLE channels and quantify continuous BLE scanning's real power cost. The authors reached experiment results with an accuracy of less than 2.6 m 95% of the time, compared with *Wi-Fi*-based implementation with less than 8.5 m 95% of the time. Additionally, the authors concluded, that positioning accuracy grows with the number of BLE beacons deployed up to a threshold of 8 to 10 beacons [FH15].

Martins et al. reach similar conclusions, but also presents a possible combination between *fingerprinting* and the *proximity sensing* techniques [Mar+20]. By creating fingerprints at specific distances from beacons, the *fingerprinting* technique outputs the estimated distance between a device and a beacon, generating a *proximity metric*. Furthermore, the authors' results deduce that the *fingerprinting* technique has 13.2% better accuracy than *trilateration* for distances up to 10 m.

Blasio et al. presents a hybrid solution for a harsh environment, a bus station, where *Wi-Fi* is used for rough positioning and *Bluetooth* for fine positioning [Bla+17]. The authors opted for *fingerprinting* as the main positioning algorithm, combined with the *Euclidean* distance metric and *weighted k-nearest neighbor* as the matching algorithm. The system achieved an accuracy of 5.21 to 7 m 90% of the time on *Wi-Fi* scenarios, and 1.47 to 2.15 m 90% of the time on *Bluetooth* settings.

Although *fingerprinting* has been extensively used in indoor positioning systems, its usage carries the following disadvantages:

- Dependency on the position of the access points/beacons.
- The density of people within the indoor environment may affect the results.
- Signal attenuation caused by objects such as walls and furniture.
- Inefficiency in the construction and maintenance of the radio map.

Regarding the last point, there have been attempts to reduce the labor cost in radio map construction from the very beginning (e.g., the propagation model employed by *RADAR* [BP00] or the previously mentioned crowd-sourcing approach [Bla+17]). Zhou et al. present a more recent approach in which they employ dead reckoning to implement a fast fingerprint acquisition method that forms a sparse radio map [Zho+18]. They then use Radial Basis Function (RBF) interpolation to extend it, with the purpose of constructing a radio map with higher fingerprint granularity.

By using the radio map constructed in this way to carry out experiments, the cumulative error probability was within 3 m 78% of the time and the average positioning error was 2.2 m. This is not as good as other standard methods of building the radio map but it is still acceptable for many applications and it can save more than 80% usually needed to build a radio map.

2.3.7.2 RSSI-based proximity, ranging and lateration

RSSI values can be used to estimate the distance between the sender and the receiver by taking into account the attenuation of the signal as it travels. *Trilateration* can then be used to determine the target's position given enough estimates to fixed terminals.

In general, distances cannot be accurately determined by taking only into account *free space path loss* because obstacles and environmental conditions unexpectedly alter the signal strength. There are models which take some of these factors into accounts, such as the *Log-distance path loss model* [Rap01], the *Ericsson Multiple Breakpoint Model* [Ake88] and the *ITU Model for Indoor Attenuation* [Int15]. However, RSSI values can vary as much as 30 to 40 dB over distances in the order of half wavelength [Caf02] and practical measurements used to track down if devices are within a cubicle can be erroneous 33% of the time [POY01].

Many models are based on observations and measurements, i.e., they are *empirical*. These are only accurate under the circumstances for which they were designed. There are others like the Motley-Keenan [KM] and the Seidel-Rappaport models [SR92] which incorporate *deterministic* aspects, e.g., wall and floor attenuation factors. However, these obstacles are often unknown.

Some of these models have been applied to IPSs. For instance, the previously mentioned *RADAR* used a propagation model to generate the *Wi-Fi* radio map automatically. However, the results were worse than when the radio map was carefully built manually [BP00].

Wang et al. use regression analysis from experimental data to find a cubic equation that approximates the distance from the received signal strength [Wan+03]. Performance varies with how *Wi-Fi* access points are distributed, but their tests indicate a mean accuracy of 1 to 3 m with a standard deviation of about 1 m when determining the target's position using *lateration*.

Ali et al. collect RSSI values across all *IEEE 802.11* 2.4 GHz channels because they can have distinct multi-path behavior [AN07]. Similarly, Lim et al. proposed a system which periodically performs RSSI measurements between APs, building a dynamic propagation model [Lim+06]. Both systems were able to get a median accuracy within 3 m [AN07; Lim+06].

Mazuelas et al. introduces an IPS relying on *trilateration* and *proximity* in which a propagation model is designed to compute and maximize the distance accuracy between access points and a mobile station [Maz+09]. Hence, the authors use a path loss model that takes advantage of RSSI to estimate distance between access points and the station. However, since the path loss model is dependable on the indoor environment, this study presents an approach to efficiently estimate the path loss exponents. The positioning is then achieved through *trilateration*. The average accuracy was 3.97 m with a standard deviation of 1.18 m.

Rusli et al. propose an improvement to the standard *trilateration* technique by using reference points in key positions where the system is prone to multipath interference [Rus+16]. The study suggests that apart from computing the distance between a user and *Wi-Fi* access points, the *trilateration* approach includes a new *offline phase* in which reference points are chosen at crucial positions where signal interference is present. This phase is followed by manually calculating the distances from each reference point to each AP. When applying *trilateration*, the distances computed between the device and the access points are matched with the distances separating the reference points and the access points, outputting the reference point with the highest similarity. Finally, the *trilateration* technique's resulting coordinates are averaged with the matching reference point's coordinates to get the user's final position. According to the authors, this improvement allows for *trilateration* accuracy to increase from two meters to one meter.

Bluetooth may provide more accurate positioning thanks to its shorter range, at the expense of higher infrastructure requirements due to a smaller coverage area per transmitter [HSK04]. For instance, Figueiras et al. built a propagation-based indoor positioning system around *Bluetooth* making use of RSSI values and obtained less than 3 m of error 90% of the times [FSK05].

Feldmann et al. present another early work in which they tackles the development of an RSSI-based positioning system by using the *trilateration* method combined with the least square estimation to predict the position of a mobile terminal [Fel+03]. Although the system achieved an accuracy of 2.08 m in a small room, the study fails to tackle *Bluetooth*-related shortcomings in indoor positioning systems as signal attenuation and multipath fading.

There is also BLE which offers lower power consumption and fast update rates [SK13]. Typically, dedicated BLE beacons periodically broadcast signals to be detected by nearby devices. They can run for several months on a button cell battery. There are multiple beacon protocols implemented by BLE beacon vendors. The most common are: *iBeacon* [App18] and Google's *Eddystone* [Goo18a].

Rida et al. analyze the performance of a *trilateration*-based positioning system which uses BLE [Rid+15]. The system determines the position of a user by detecting the three nodes with the highest RSSI when a device enters the broadcast range. Experiments with the system resulted in an average of 0.5 to 1 m of accuracy.

SpotON uses RSSI values to estimate distances between RFID readers and tags. Positioning has an accuracy of about 3 m but it takes 10 and 20 s [HBW00]. *LANDMARC* uses active RFID tags to achieve errors of around 1 m, but it also suffer from long processing times [Lio+04].

The examples presented so far often use signal strength to estimate distance to fixed points in the environment and then use *lateration*, or other applicable technique, to determine the absolute position of a target device. Therefore, most of these systems can also be modified to simply provide proximity information to those reference points. For instance, Martins et al. and Rida et al. present possibilities for such solutions providing different distance calculations that enable *proximity* approaches [Mar+20; Rid+15].

Focusing solely on the *proximity* technique, Klokmose et al. propose a *Wi-Fi proximity detection* system for mobile web applications based on proximity-adaptive HTTP responses [KKB14]. Their product, *ProxiMagic*, adapts the response to an HTTP request to a web server by changing the web application's interface according to the relative position between the user's device and a set of PoI. Despite the arguments of usability and cost-effectiveness of the system, *ProxiMagic* fails to provide proximity information when the device is not generating network traffic and is prone to inadequate proximity events since the system only uses an RSSI threshold to trigger proximity events.

Regarding *proximity* approaches using BLE technology, Zafari et al. propose a solution to the lack of accuracy in *iBeacon*'s PBS approach, specifically in proximity classification [Zaf+17]. By providing two server-side algorithms leveraging the moving average and *Kalman* filtering techniques, the authors tackle the shortcomings of *iBeacons* both in environments with less interference noise like coffee shops or small stores and environments with more significant interference noise. The experiments show that the two algorithms' employment improves *iBeacon*'s *proximity detection* by 29% and 32%, respectively.

Kim et al. research also tackles the same shortcoming in *iBeacon*'s *proximity detection* accuracy by providing a solution based on time window, RSSI frequency, and user's walking [Kim+15]. The authors developed a proximity zone detection technique and algorithms within a 0.5 m radius range that takes advantage of step detection, filtering techniques such as moving average with a specific window size, and the RSSI's frequency to achieve an accuracy improvement of up to 12.6% when compared with existing techniques.

Kalbandhe et al. present a solution where BLE beacons are deployed throughout an environment to provide indoor positioning estimations [KP16]. The tags are configured to transmit advertising packets at −76 dBm continuously and are deployed throughout the environment. When a user enters the range of such tags, the application, previously installed on their mobile devices, extracts the RSSI value which is used to estimate the distance between the tags and the mobile device. In terms of performance, the authors state that this approach remains relatively accurate up to distances of 4 m.

To build a robust system capable of using positioning techniques that fit the environment, decision systems have been a research subject in indoor positioning systems. For instance, Orujov et al. propose a *fuzzy logic* scheme to tackle changes in a environment [Oru+18]. The *fuzzy logic* scheme is used to select the most appropriate indoor positioning technique, including *trilateration* and *proximity*, based on the conditions that mostly differ: number of BLE beacons, size of rooms, and signal strength.

Moreover, their research investigates the broadcasting range of BLE beacons in a real-world environment. The authors suggest that the maximum range of the signal broadcast in real environments does not match the manufacturer's range. For instance, instead of 50 m as stated by *Estimote*, the real maximum range was only 10 m. They also concluded that for an efficient operation of indoor positioning algorithms, the data obtained from beacons has to be from within a 3 m range.

2.3.7.3 Infrared-based systems

Infrared is electromagnetic radiation with wavelengths longer than those of visible light. It is therefore invisible to the human eye, which means that infrared signals can unobtrusively be used as part of an indoor positioning system. Infrared-based positioning systems are typically composed of an infrared light-emitting diode, responsible for the transmission of an infrared signal as non-visible light, and a receiving photodiode to detect and capture the light pulses [Bre+17].

The *Active Badge* system is one of the most well known infrared-based positioning systems. It uses ceiling mounted sensors to detect infrared signals emitted by badges carried [Wan+92] (Figure 2.13(a)). Sensors are polled by a central server and each detected badge is assumed to have the same position as the sensor that detected it. The accuracy of the system is thus limited by the density of the sensors.

In a similar system called *Wireless Indoor Positioning System (WIPS)*, the infrared signals are emitted by transmitters mounted in the environment [KTH00] (Figure 2.13(b)). The signals are received by badges to determine their position. The badges need to be more complex than the ones used in *Active Badge* because they incorporate an infrared sensor for positioning and WLAN for connectivity with a central server. However, this approach provides better privacy because the badge does not expose its location by continuously broadcasting a signal.

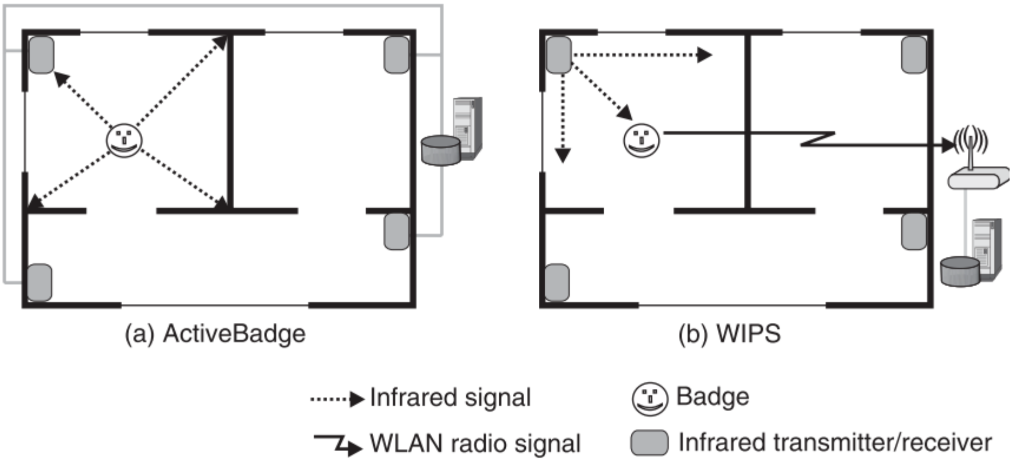


Figure 2.13: *Active Badge* and *WIPS* [Küp05]

Light and infrared-based positioning systems present some issues [AI11; Küp05; KPV11]. Line-of-sight is required and each sensor only provides a limited coverage area, which limits the system scalability to large environments because of the cost of the required infrastructure. They can also be susceptible to interference from sunlight or fluorescent light. Moreover, depending on the system, users may have to continuously wear a visible badge.

2.3.7.4 Visible Light

Visible Light Communication (VLC) uses visible light to transmit data using artificial light sources such as LED lamps. The principle behind is ability to switch lights on and off in intervals so fast that the human eye cannot perceive it [Bre+17]. Moreover, the potential use of existing artificial light infrastructure, and the ambient light sensors or cameras present on smartphones and COTS devices, make this technology an attractive solution for indoor positioning systems in terms of costs. Despite these advantages, the reliances on visible light sources makes VLC system dependent on the arrangement of the available light sources.

The principle for VLC is that each of the fixed lamps has different flicker encoding. A sensor, which could be carried by the user, receives the light and compares the modulation against the known encoding schemes and determines which is the dominant one, thus associating the sensor location with the vicinity of the corresponding lamp (see Figure 2.14).

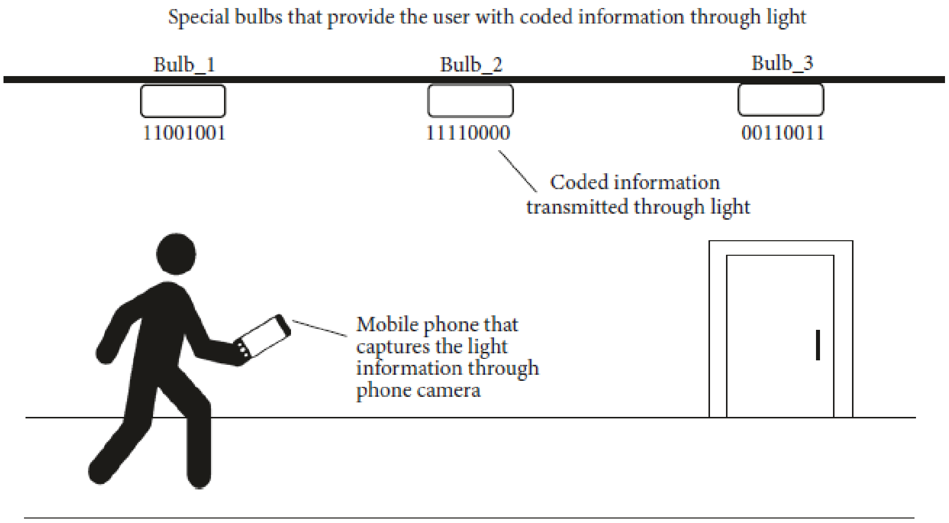

Special bulbs that provide the user with coded information through light
Bulb_1
Bulb_2
Bulb_3
11001001
11110000
00110011
Coded information transmitted through light
Mobile phone that captures the light information through phone camera

Figure 2.14: Indoor Positioning System based on *Visible Light Communication* [Bre+17]

Yasir et al. present an IPS that uses VLC and accelerometer measurements to calculate the distance between transmitters (LED light sources) and receivers (mobile device equipped with a photodiode and an accelerometer) [YHV14]. By taking into account both distance, acceleration and orientation, the proposed system reached an average positioning accuracy of less than 0.25 m. Zhang et al. also proposed a system with reportedly less than 0.2 m of accuracy 95% of the time [ZCK14].

2.3.7.5 Ultrasound-based systems

Sound waves propagate much slower than electromagnetic waves, which means that time measurements and synchronization mechanisms can be less accurate. Line-of-sight between sender and receiver is also not required [Küp05]. Ultrasounds are especially interesting because they are inaudible, i.e., their frequency is above the 20 kHz threshold of human hearing. Moreover, their propagation range is limited, so they tend to only cover a room [Wer14].

The *Active Bat* is a well known indoor positioning system that uses ultrasounds. It consists of a sensor network of ultrasound receivers on the ceiling [Har+99a; WJH97]. A transmitter broadcasts an ultrasound signal after receiving an RF signal from a central node. The time it takes for the ultrasound signal to be detected by the receivers is used to estimate the transmitter's position through *trilateration*. This works because the RF signal travels nearly instantaneously when compared to the speed of sound. The system has an accuracy of 9 cm 95% of the time and it can be extended to support orientation determination [Har+99a].

Cricket works similarly but the badges are equipped with a radio and an ultrasound receiver [PCB00]. Ceiling mounted beacons emit a radio signal and an ultrasound impulse simultaneously. The time difference between the arrival of the two signals can be used to measure the distance between a beacon and the badge. This approach provides better privacy, because position determination is made by the terminal instead of the network.

The *Cricket* system was initially designed to only determine the nearest beacon following a *proximity sensing* approach, but it was eventually extended to support *lateration* [Bal+03]. *Cricket Compass* adds orientation information [Pri+01] and *Cricket v2* can reach an accuracy of 1 to 2 cm with the correct beacon density [Bal+03; Bal+05]. Unfortunately, these systems require users to carry a dedicated device and infrastructure requirements grow with coverage area. They may also be vulnerable to collisions between the signals emitted by tags if a robust scheduling scheme is not implemented.

2.3.7.6 Acoustic-based systems

There have also been research into positioning solutions that use audible sound instead of ultrasounds. This has the advantage of allowing the repurpose of hardware that is already available and part of many devices (e.g., speakers and microphones). However, there is the downside of the acoustic signals being potentially heard by users. Nevertheless, it should be possible to attenuate this by keeping the signals very short and by using certain frequencies.

Regarding examples of research in this area, Oh et al. developed a relative positioning system using acoustic signals without relying on specialized hardware [OLL09]. It is based on another system called *BeepBeep* and eliminates the need for clock synchronization between devices by having each device record their own acoustic signals [Pen+07]. Relative positions can actually be more meaningful when trying to make sense of our surroundings [Gel+08].

These systems rely on COTS devices using commodity and readily available hardware on smartphones, i.e., a microphone and a speaker. *BeepBeep* employs an Elapsed Time between two Times of Arrival (ETOA) approach in which each device would alternately emit a *beep* sound signal (two-way sensing phase), while also using its microphone to record its own signal (self-recording phase). Moreover, since each recording would have two *beep* signals, the system would calculate the number of sound samples between the two signals and divide it by the sampling rate (sample counting phase). These calculations would derive the ETOA between the two signals emitted, which was then applied to compute the distance between the two devices. The experimental results assessed an accuracy down to the centimeter level, with less than 2 centimeters of standard deviation.

The probability of collisions between acoustic signals and computational time increases with the number of devices [OLL09]. As such, Oh et al. proposed a protocol to schedule the transmission of the acoustic signals and introduced optimizations to reduce computation time by over 83%. The system has an accuracy of 12 cm 90% of the time, whereas the unmodified *BeepBeep* system has a 7 cm accuracy 90% of the time [OLL09]. The loss of accuracy is due to the analysis of shorter audio samples and coarser grained calculations.

RoomSense follows a different approach. It relies on sound *fingerprinting* in which rooms and within-rooms positions are characterized by the Maximum Length Sequence (MLS) impulse response [Ros+13]. A Support Vector Machine (SVM) model was trained to classify the rooms and within-rooms positions which can then be used during the positioning phase to estimate positions. According to the authors, *RoomSense* reaches an accuracy of 98 % on a room level and 96 % within-rooms.

2.3.7.7 UWB

UWB is a radio technology that transmits short-range radio waves formed by a sequence of very short pulses using a high-bandwidth (more than 500 MHz). A typical UWB setup includes radio wave generators and receivers that capture the propagated and scattered waves. It distinguishes itself from other technologies, such as *Bluetooth* and *Wi-Fi*, by holding the properties of strong multipath resistance and penetrability for building materials [Mau12]. These features, aligned with the high precision associated with UWB, have led to this technology's growth in indoor positioning research.

Among the research made in this scope, the work of Bai et al. proposes an IPS based on UWB technology, in which four fixed transmitters send UWB signals to mobile users [YX09]. Through the method of TDOA, the system is able to estimate the location of the users. According to the authors, the built solution reaches an accuracy of less than 1 m when the UWB signal bandwidth is 528 MHz.

There are also commercial and industrial products in this area. For instance, *Estimote* has incorporated UWB into their advanced *AI Beacon* product and promise centimeter level accuracy [Est21]. Other solutions include products by *Pozyx* and *Decawave*, among others [Dec21; Poz21]. *Apple* and Samsung are also now starting to add UWB capabilities to some of their products. If this trend continues, with UWB becoming widely available as part of consumer grade products, it may be worth to focus some of our future research on this technology.

2.3.8 User Identification and Tracking

Positioning systems are primarily focused on improving accuracy and precision. However, it is also important identify and track users as part of proxemic-based interactions, with the smartphone being often used for that purpose. For instance, on *Wi-Fi fingerprinting*-based systems, smartphones often have take the role of collecting *fingerprints*. Their sensors may also be used to gather further contextual information as part of a sensor network [SK13].

However, contrary to common perception, people do not keep their smartphones in close proximity most of the time. Real world studies have found that they tend to only have them turned on and within arm's reach about 50 % of the time. The study also found that users are more likely to keep their phones at a distance when at home [SK13]. Therefore, tracking smartphones down to a few centimeters may be pointless when it is the actual user that we are interested in.

After realizing that the relationship between users and their smartphones is weaker than expected, it becomes important to push forward the use of wearable devices as stronger personal identifiers. This may be achieved through small dedicated signal transmitters carried by users [SK13], but it constitutes an additional barrier of entry. However, with the rise in popularity of smartwatches and fitness trackers, it may be possible to leverage them for positioning.

3

Framework for Pervasive Cross-device Applications

YanuX is the name given to our framework for cross-device applications running across co-located. This chapter summarizes the framework's high-level architecture, presents its components and details the solution for the UI component distribution.

3.1 Framework's Architecture

Figure 3.1 shows a diagram of the *YanuX Framework*'s architecture. The three components aligned at the bottom are server-side components (*IPS Server*, *YanuX Auth*, *YanuX Broker* and *YanuX IPS Bridge*). They only need to be hosted on an network that is accessible from the client devices, but ideally they should be deployed on publicly accessible hosts so that they can be accessed through the Internet. Moreover, the *IPS Server* is depicted because it was integrated with the rest of system thanks to the *YanuX IPS Bridge*. However, the Indoor Positioning System (IPS) is a standalone contribution presented in Chapter 4.

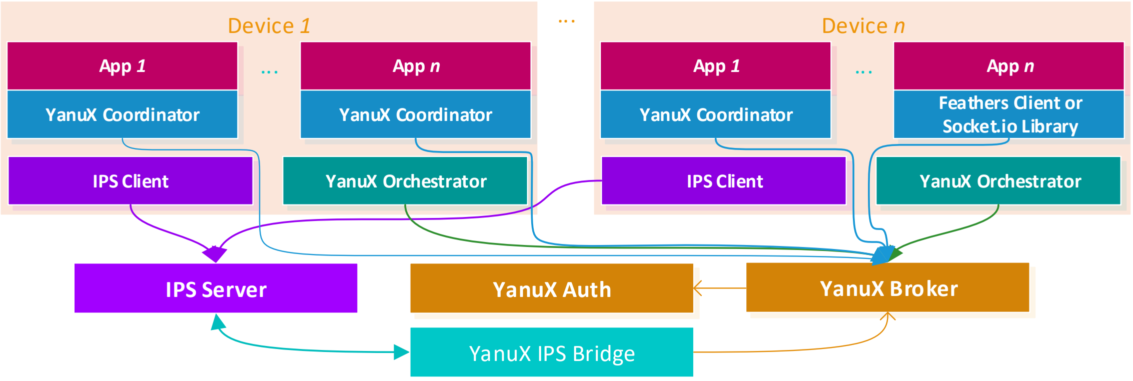


Figure 3.1: *YanuX Framework* Simplified Architecture

The multiple devices that are configured to run *YanuX Framework*-based applications appear on top of the architecture diagram. Each device can host multiple applications

(*Apps*), which access the framework's services through a library (*YanuX Coordinator*) that abstracts some of the lower level communication details while also taking care of the component distribution decisions. Alternatively, devices may also communicate with the *YanuX Broker* by establishing a direct connection using one of the libraries mentioned in Figure 3.1 below *"Device n/App n"*.

IPS Client and *YanuX Orchestrator* are two components shared by all applications running on one device. Only a single instance of each of these components is executed in the background on each device. Moreover, just like the *IPS Server*, the *IPS Client* is part of the standalone Indoor Positioning System (IPS) presented in Chapter 4. As for the arrows connecting the components, a single-sided arrow means a request is made from the originating component to the target component. Double-sided arrows represent that a bidirectional channel is kept open between the two components once the communication is established from a client component (on one of the devices) to a server component.

More information, documentation and the repositories for all of the components of the framework can be found at: `https://yanux-framework.github.io/`.

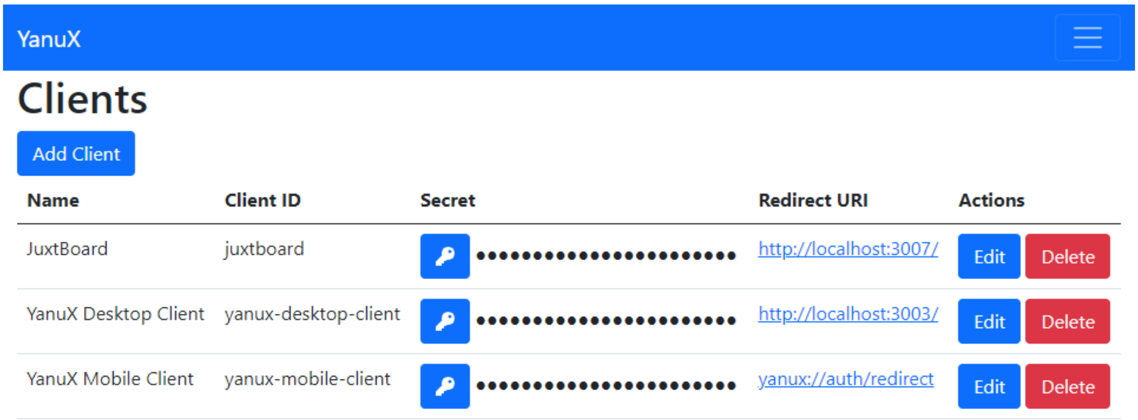
3.2 Framework's Components

This section presents the core components of the *YanuX Framework*, which are detailed as follows.

3.2.1 YanuX Auth

In order to correctly store data about the current state of an *application* created by a certain *developer* that is being used by a given *user*, we need to somehow keep track of the *applications*, *developers* and *users* that are using the *YanuX Framework*. Therefore, *YanuX Auth* provides authentication and authorization services. It also provides a website which anyone interested in developing an application based on *YanuX* must first sign up to register their client applications (see Figure 3.2). Meanwhile, users must log in and give authorization to each application that they want to use.

YanuX Auth implements an *OAuth 2.0 Authorization Server* [D H12] with Proof Key for Code Exchange (PKCE) (Proof Key for Code Exchange) [BA15] support that issues access tokens used by the client applications to make requests to other components on each user's behalf [D H12]. This way, users do not have to directly provide their credentials to applications, and they can separately approve and revoke the access given to each application. A *Resource Server* was also implemented to expose a service that can be used by other components to check the validity of an access token, as well as a standard *OpenID Connect* endpoint [Sak+14].



YanuX

Clients

Add Client

Name	Client ID	Secret	Redirect URI	Actions
JuxtBoard	juxtboard	•••••••••••••••••••••••	http://localhost:3007/	Edit Delete
YanuX Desktop Client	yanux-desktop-client	•••••••••••••••••••••••	http://localhost:3003/	Edit Delete
YanuX Mobile Client	yanux-mobile-client	•••••••••••••••••••••••	yanux://auth/redirect	Edit Delete

Figure 3.2: *YanuX Auth* Client Application Management Page

3.2.2 YanuX Broker

Since an application runs on multiple devices, it is imperative to keep track of its global user interface state and of any proxemic relationships established between the devices present in the users' surrounding environment. Therefore, the *YanuX Broker* exposes *JavaScript Object Notation (JSON)* (JavaScript Object Notation) objects which should be used by applications to store the global UI state. The application instances on each device can then manipulate and subscribe to changes made to those objects.

Within the context of our framework, those objects are called *resources*. Initially, there was a single *resource* for each user and client application combination. However, the framework has been extended so that each user can have multiple *resources* stored for each application. It is also possible to share *resources* with other users to enable collaboration.

Resources should be used to store the complete UI state of the client application. Application instances on each device can then manipulate and subscribe to any changes made to the object belonging to the current authenticated user. This is more efficient than periodically polling the server for change and allows a more responsive UI since changes should be reflected almost instantaneously across devices.

There may be some types of changes made to the UI that do not translate well to a global change in the UI state and the corresponding re-rendering of the client application instances on every device. So the framework also supports the submission of events to the *YanuX Broker*. Each event should have an associated name and payload (a *JSON* object).

Once received by the *YanuX Broker*, the event will automatically be sent to any subscribing instances of the same client application and user as the one that submitted it. In contrast with the global UI state object, these events are ephemeral. They are simply received and broadcast. They are not stored for future usage, i.e., there is no way for a client application instance to know about events that have been previously received and dispatched.

The *YanuX Broker* is also responsible for aggregating data coming from the *YanuX IPS Bridge* into per-user proxemic states and for keeping subscribing application instances up-to-date. The proxemic states can be fed to the *YanuX Coordinator* to determine which UI elements should be shown. When a user is subscribed to a shared *resource* its proxemic state is shared with the same users that the *resource* is shared with.

Proxemic state changes can then be fed into the *YanuX Coordinator* library as part of the process of determining which UI elements should be visible based on the current proxemic relationships between co-located devices. If developers do not wish to use our implementation they are free to implement their own logic based on the same proxemic state data.

All of these services are exposed through a Representational State Transfer (REST) API: `https://yanux-framework.github.io/YanuX-Broker/`. However, the same set of services are also made available to *Feathers* [Fea18] or *Socket.io* [Soc18] compatible clients, which allow the server to push events to connected clients as needed.

3.2.3 YanuX Coordinator

The services provided by the *YanuX Broker* can be directly accessed using any of the previously mentioned options. Therefore, developers are free to implement application instances on any programming language, platform or OS they desire. However, we have also written a helper library called *YanuX Coordinator* using *TypeScript* [Mic17]. The library makes it easier to access the *YanuX Broker* from Web-based applications or hybrid applications, e.g., by using a *WebView* [Goo18b] on an *Android* application or a desktop application built with *Electron* [Git18].

The library takes care of sending the correct messages to the *YanuX Broker* for client authentication and event subscription so that third-party developers can focus on building their own applications, without needing to be aware of the lower level details of the *YanuX Framework*. Developers only need to provide the correct authentication token, which they have acquired previously from *YanuX Auth*, and to register callback functions for certain types of events that the server can push to connected clients.

Those events include: changes to the object with the global UI state or to the proxemic relationships between devices; notifications of when a new device becomes active or inactive; and developer defined custom events. Depending on the type of event, developers may program their applications to simply change a minor part of the UI or completely re-render it according to the new information that they just received.

The API of the *YanuX Coordinator* can be summarized as follows:

- **constructor(brokerUrl, localDeviceUrl, clientId, credentials)**: The constructor that creates a *Coordinator* object with the details needed to connect to the *YanuX Broker*.

- **Properties**:

- **client**: Client application that is connected to the broker.
- **device**: Device running the current application instance.
- **instance**: Local instance of the application.
- **proxemics**: Proxemic relationships of the devices the user has access to, i.e., including those that are available via resource sharing with other users.
- **resource**: Default resource of the user on the current app.
- **subscribedResourceId**: Id of the subscribed resource.
- **user**: The user using the current application instance.

- **Methods** (arguments with *?* have a default value):
  - **clearComponentsDistribution(event)**: Helper method that clears the distribution of components by calling *setComponentDistribution* based on *reset-auto-components-distribution* event generated by the *ComponentsDistributionElement*.
  - **createResource(resourceName?)**: Creates a new resource.
  - **deleteResource(resourceId)**: Deletes a resource.
  - **distributeComponents(event)**: Helper method that distributes components by calling *setComponentDistribution* based on the *updated-components-distribution* event generated by a *ComponentsDistributionElement*.
  - **emitEvent(value, name)**: Emits an application event.
  - **getActiveInstances()**: Gets active instances that the current user has access to, i.e., including those that are available via resource sharing with other users.
  - **getInstances(extraConditions?)**: Gets instances that the current user has access to, i.e., including those that are available via resource sharing with other users.
  - **getProxemicsState()**: Gets the proxemic relationships of the devices the user has access to, i.e., including those that are available via resource sharing with other users.
  - **getResourceData(resourceId?)**: Gets a resource's data.
  - **getResources()**: Gets the resources that the user currently has access to, i.e, including shared resources.
  - **init()**: Initializes the connection to the broker.
  - **isConnected()**: Indicates if the Coordinator is connected.
  - **logout()**: Logs out from the Broker.
  - **setComponentDistribution(components, auto?, instanceId?)**: Sets the distribution of UI components.

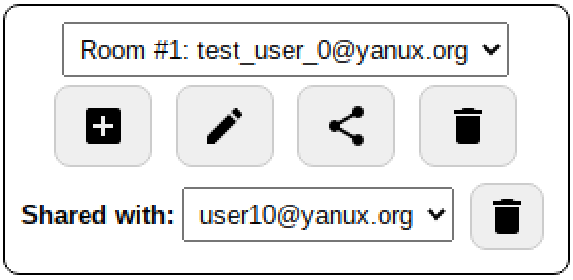
- **setInstanceActiveness(active)**: Sets if the current instance is active, i.e., if it is being used.
- **setResourceData(data, id?)**: Sets a resource's data.
- **shareResource(userEmail, resourceId?)**: Shares a resource with a user.
- **subscribeEvents(subscriberFunction)**: Subscribe to application events.
- **subscribeInstances(subscriberFunction)**: Subscribe to changes made to instances.
- **subscribeProxemics(subscriberFunction)**: Subscribe to changes made to the proxemics that user has access to, i.e., including those that are available via resource sharing with other users.
- **subscribeReconnects(subscriberFunction)**: Subscribe to reconnection events.
- **subscribeResource(subscriberFunction, resourceId?)**: Subscribe to changes made to a resource.
- **subscribeResourceSubscription(subscriberFunction)**: Subscribe to changes made to the resource subscription, i.e., which resource is subscribed by the user.
- **subscribeResources(subscriberFunction)**: Subscribe to changes to made to the resources that a user has access to, i.e., owned or shared with.
- **unshareResource(userEmail, resourceId?)**: Unshares a resource that was shared with another user.
- **unsubscribeEvents()**, **unsubscribeInstances()**, **unsubscribeProxemics()**, **unsubscribeReconnects()**, **unsubscribeResource()**, **unsubscribeResourceSubscription()**, **unsubscribeResources()**: Unsubscribe from the corresponding subscriptions mentioned above.
- **updateComponentsDistribution(componentsRuleEngine configureComponents, componentsDistributionElement?, instanceId?, ignoreManual?)**: Update the distribution of UI using a *ComponentsRuleEngine* instance. The *configureComponents* will be called with the new distribution. A *ComponentsDistributionElement* can be passed so that it gets automatically updated.
- **updateInstanceActiveness()**: Determines if the current instance is being used and sets its activeness accordingly.
- **updateResources(resourceManagementElement?)**: Gets the updated resources that the current user has access to. If a *ResourceManagementElement* is passed, it gets updated with the most recent resource information.

More details are available online at the following Uniform Resource Locator (URL): `https://yanux-framework.github.io/YanuX-Coordinator/interfaces/coordinator.html`.

3.2.3.1 Resource Management

The framework allows users to store multiple application states (*resources*) per application. This can be useful for multi-tasking, e.g., multiple opened documents, seamlessly switching back and forth between multiple videos, or participating in multiple conversations.

As previously stated, the framework allows sharing *resources* with multiple users. This allows the creation of cross-device applications with many users collaborating in the same space. For instance, we can envision collaborative cross-device document editors, classroom applications or digital canvases. Therefore, we have developed a *Web component* named *YanuX Resource Management Element* to assist developers with the management of those multiple *resources* in an application's UI (see Figure 3.3).



Room #1: test_user_0@yanux.org

Shared with: user10@yanux.org

Figure 3.3: The *YanuX Resource Management Element* provided by *YanuX Coordinator*.

The list of *resources* owned by or shared with a user are fed into the *Web component*. They are listed at the top of the component in a drop-down list which shows the resource name and the owner's email. An event is triggered when users select a resource in the drop-down list so that developers know which resource they should instruct the *YanuX Coordinator* to retrieve and subscribe to.

The buttons below the drop-down list allow users to create a new resource, to rename the currently selected resource, to share it by providing a user's e-mail address or to delete it, respectively. There is a second drop-down that will be populated with the e-mail of the users with whom the resource has been shared. The owner will also be able to click on the button next to the list to stop sharing the resource with the currently selected user.

All of these user actions will show a dialog box asking for extra input and/or confirmation. Once confirmed, the *Web component* fires the proper events which let developers know that they should make the relevant *YanuX Coordinator* API calls to actually perform the operations locally and at the *YanuX Broker*.

The *YanuX Resource Management Element* API can be summarized as follows:

- **HTML Attributes**:
  - **resources**: The resources that the user has access to, , i.e, including shared resources.
  - **selectedResource**: The currently selected resource.

  - **selectedResourceId**: The Id of the selected resource.
  - **userId**: The Id of the current user.

- **Custom Events**:
  - **create-resource**: The *detail* property contains the *resourceName* of the created resource.
  - **delete-resource**: The *detail* property contains the *resourceId* of the deleted resource.
  - **rename-resource-name**: The *detail* property has the renamed *resource*, its *resourceId* and its new *resourceName*.
  - **resource-selected**: The *detail* property contains the selected *resource* and its *resourceId*.
  - **share-resource-email**: The *detail* property contains the *resource*, its *resourceId* and the *userEmail* of the user with whom the resource was shared.

More details are available online at the following URL: `https://yanux-framework.github.io/YanuX-Coordinator/classes/resourcemanagementelement.html`.

3.2.3.2 Component Distribution

Another key aspect of the *YanuX Coordinator* is its component distribution approach based on the capabilities of each device and the requirements of each UI component. Although this aspect is absolutely optional, it should ease the burden of developing something similar on a case by case basis while providing a flexible system to build engaging experiences.

The *YanuX Orchestrator* runs in the background and is responsible for gathering information about each device's capabilities. However, it is also possible to override that information or to add additional details that the *YanuX Orchestrator* is incapable of determining by itself.

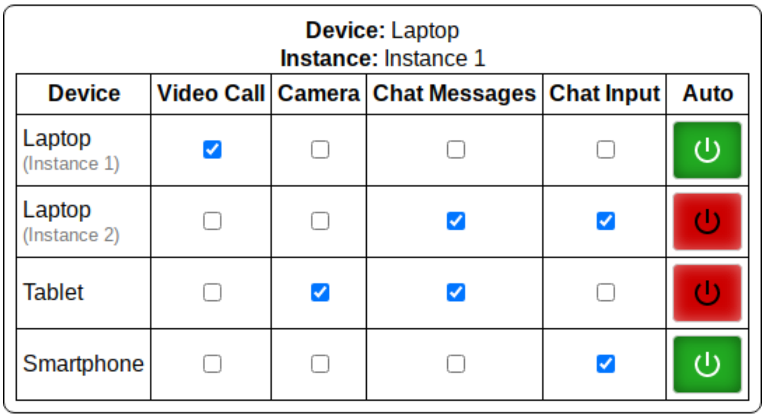
We are using the term *capabilities* to broadly encompass any of the input/output characteristics of the current device or even its context of use, e.g., a public display could have video output, but no way to directly receive user input or to provide audio output, while it should probably not be used to display private information. This is the type of information that is sent to and stored by the *YanuX Broker* (check *YanuX Orchestrator*'s description for further details).

Another piece of information that the *YanuX Broker* keeps track of is the list of connected instances for each application. With the assistance of the *YanuX Coordinator* it is even able to determine if an instance is active or not, i.e., in a web application an instance inactive whenever it is connected to the *YanuX Broker*, but the browser is minimized or a different tab is focused.

Developers need to add a *JavaScript* object to their application that follows our *JSON-based Domain-specific Language (DSL)*, as defined here: `https://yanux-framework.github.io/docs/components-restriction-language/schema.json`. The name of the properties of the object are the unique names of each UI component of the application. For instance, on a web application one may use a unique *id* for each *div* element which encapsulates an UI component.

To better understand our approach, we present a practical example of the restriction definition *DSL* and more information about the distribution algorithm itself when we present our prototypes in 5. There is also documentation about the DSL schema available at the following URL: `https://yanux-framework.github.io/docs/components-restriction-language/schema_doc.html`.

The automatic component distribution approach should strive to come up with the best solution. However, users should be able to override it because they might have different needs than the ones assumed by the framework or the application's developers. Therefore, we have created a *Web component* named *YanuX Components Distribution Element* that allows developers to give the control over the distribution of UI components back to users (see Figure 3.4).



Device: Laptop
Instance: Instance 1

Device	Video Call	Camera	Chat Messages	Chat Input	Auto
Laptop (Instance 1)	☑	□	□	□	🟩
Laptop (Instance 2)	□	□	☑	☑	🟥
Tablet	□	☑	☑	□	🟥
Smartphone	□	□	□	☑	🟩

Figure 3.4: The *YanuX Components Distribution Element* provided by *YanuX Coordinator*.

The names of the device and of the instance where the element is placed are displayed at the top. However, the name of the instance will only be shown if there is more than one instance on the same device. In the example, the table shows a total of four applications instances running on three distinct devices (2nd to 5th rows) and four components (2nd to 5th columns). The name of the instances is only shown if there is more than one instance per device.

The header of the 2nd to 5th columns displays the name of each of the components while the rows below contain check boxes whose selected states reflect the current distribution of components across the active application instances. Users can change the distribution by checking/unchecking if a given component should be shown on the corresponding application instance listed in the 1st column. When that happens, the *web* *component* emits an event.

The 6th column has buttons that indicate if the distribution for a certain application instance was determined automatically (pressed green state, like in the 2nd and 5th rows) or if it was modified by the user (unpressed red state, like in the 3rd and 4th rows). All buttons start in the pressed green state because the distribution of the components is automatically determined by default.

When a user changes the components distribution the corresponding button will become unpressed and red. Moreover, when a user presses one of those buttons the *YanuX Components Distribution Element* generates an event signaling that the user intends to reset the components distribution of the corresponding instance to be determined automatically, hence the column header *Auto*.

The *YanuX Components Distribution Element* API can be summarized as follows:

- **HTML Attributes**:
  - **componentsDistribution**: Distribution of UI components.
  - **instanceId**: The Id of the current instance.
- **Custom Events**:
  - **reset-auto-components-distribution**: The *detail* property contains the *instanceId* that had its distribution reset.
  - **updated-components-distribution**: The *detail* property contains the *instanceId*, the name of the *component*, the *checkboxChecked* status and the updated *componentsDistribution*.

More details are available online at the following URL: `https://yanux-framework.github.io/YanuX-Coordinator/classes/componentsdistributionelement.html`.

The *YanuX Coordinator*'s distribution algorithm should be invoked whenever there are changes in the proxemic relationships of co-located devices and in the available instances, when there is a manual change in the components distribution, or when a users requests to reset the distribution of an instance.

The algorithm will run on each of the co-located devices and independently determine whether each UI component should be available on that device. The algorithm will always give priority to the distribution made by the user. When no manual distribution is defined, it will automatically determine one based on the restrictions defined by the developer, the current proxemic relationships between devices, whether each current application instance is active, and the capabilities of each device running those instances.

Each device may also run the component distribution algorithm for other devices that are present and active whenever a certain restriction is just a preference instead of a requirement. In such a case, the algorithm will first check if the device fulfills the restriction. If not, it will check if any of the other devices are able to fulfill it. If they are able to do so, the algorithm will treat the restriction normally as unfulfilled. Otherwise, the algorithm will consider that the requirement has been fulfilled. This may lead to a component being enabled that would otherwise be disabled.

The end result of the distribution algorithm will be a *JavaScript* object with the value *true* or *false* for each of the components defined in the restrictions. It is up to the developer to interpret these values. The most direct approach on a web application is to set the *display* property in *CSS* to *block* for all components with *true* returned by the algorithm and *none* for the remaining ones.

The API of the *ComponentsRuleEngine* that implements our distribution algorithm can be summarized as follows:

- **constructor(localInstanceUuid, localDeviceUuid)**: Initializes some properties of the distribution engine.
- **Properties**:
  - **currentComponentsDistribution**: Current distribution of UI components.
  - **instances**: Instances considered by the algorithm.
  - **localDeviceUuid**: Universally unique identifier (UUID) of the local device.
  - **localInstanceUuid**: Id of the local instance.
  - **proxemics**: Proxemic relationships of the devices the user has access to, i.e., including those that are available via resource sharing with other users.
  - **restrictions**: *DSL* that imposes restrictions on the distribution of components.
- **Methods**:
  - **run(ignoreManual?)**: Runs the distribution algorithm. *ignoreManual=true* resets the distribution to automatic.

More details are available online at the following URL: `https://yanux-framework.github.io/YanuX-Coordinator/classes/componentsruleengine.html`.

3.2.4 Indoor Positioning System (IPS)

The framework needs to have an updated perspective of the devices available in the surrounding environment. It relies on the awareness of the devices' proxemic relationships, specifically the distance, orientation, and identity dimensions [Gre+11]. Since there is no readily available solution that enables us to capture that information, we had to develop our own.

The *IPS Client* is available for *Android* and *Linux* (it should also run on *Windows* and *macOS* if the operating systems are configured correctly). It scans for *BLE* (Bluetooth Low Energy) beacons and *Wi-Fi* access points, and sends its findings to the *IPS Server* along with the orientation reported by the device's sensors. The server estimates the device's position and publishes the results to a Web Application Messaging Protocol (WAMP) running besides it [WAM21]. Any interested party can subscribe to those position updates.

The *IPS Server* supports radio map *fingerprinting* and *proximity sensing* techniques that apply machine learning algorithms to predict a device's absolute position in an area and the relative position to *BLE* beacons, respectively. *Trilateration* is also supported and it applies least-squares optimization to estimate a device's position based on the distance to the known location of at least three *BLE* beacons.

The server uses *fuzzy logic* to decide what is the most appropriate positioning technique in a specific situation. By following this procedure, the system's performance is expected to be at least equal to the performance achieved by the best performing algorithm [Oru+18].

During the user studies presented on Chapter 6, we used the *IPS Server* in *proximity sensing* mode, i.e., the estimates of the distances between devices were used directly to aggregate devices instead of inferring those distances from absolute positions. The devices used the *IPS Client* to scan for and emit BLE beacon signals. The PC would use a physical beacon near the display instead of emitting its own signal. This decision was taken to simplify the deployment of our test environment, because building a radio map for *fingerprinting*, or placing and calibrating *Bluetooth Low Energy* beacons for *trilateration*, was not required. Besides, the relative distance of the devices was all that was required for the scenarios that were being tested.

Moreover, the estimates returned by *proximity sensing* in previous experiments were more stable and accurate than relative distances calculated from absolute positions provided by the *fingerprinting* or *trilateration* methods. Therefore, this approach should lead to a better user experience in these user studies.

Please note that this Indoor Positioning System (IPS) is explained in further detail as part of Chapter 4.

3.2.5 YanuX IPS Bridge

This component communicates with the IPS and restructures position predictions to a format that can be understood by the *YanuX Framework*. It integrates the framework with the positioning system by subscribing to the *IPS Server* through the WAMP protocol [WAM21]. It tracks successive events from the same device and passes them through an *exponential moving average* filter to smooth out any errors before sending the measurements to the *YanuX Broker*.

The *YanuX Broker* is responsible for processing the data submitted and establishing proxemic relationships between devices in a given environment. These relationships are established between devices that belong to users that have access to the currently selected *resources* (i.e., both owners and users that the *resources* have been shared with), if their positions are within 3 m of each other and their orientations deviate by less than 90°. These values were chosen based on empirical experience during the development of the solution. Nonetheless, there was the rationale of 3 m corresponding roughly to the border between the *social* and *public* spaces according to the proxemics theory [Hal90], and that 90° allow us to focus on a quadrant directly in front of the users.

Thanks to the *YanuX IPS Bridge*, the *YanuX Broker* holds up-to-date information about the client devices present in a particular environment at any given moment (see Algorithm 1).

Algorithm 1 YanuX IPS Bridge Component

```
State:
   $positionTopic \leftarrow onLocationUpdate$
Upon Init() do:
   Trigger subscribe(positionTopic)

Upon NewMessageOnTopic(topic,data) do:
   If($positionTopic = topic$) do:
      $structuredData \leftarrow structureDataInKnownFormat(data)$
      Trigger SendToYanuXBroker(structuredData)
```

3.2.6 YanuX Orchestrator

One of the main issues associated with cross-device applications arises from the broad variety of devices in which they must run. The functions of an application may remain the same, but the devices that are used for running the application may have different input and output modalities [Par+18]. Screens may range from the tiny ones found on smartwatches to large TV screens, while others may not have a screen at all (e.g., smart speakers like the *Amazon Echo* and *Google Home*). Input may come from *T9* keypads, touchscreens, mouse, or physical *QWERTY* keyboards.

A single UI solution will most likely not fit the full range of possibilities [TC99]. Thus, the UI of these applications should be responsive and adapt to the set of devices where they currently run [Gal+16]. Therefore, we must first identify the different capabilities of each device. Table 3.1 presents the ones that we have considered so far.

The enumeration of the capabilities is done by the *YanuX Orchestrator*, a background service that runs on each device. It works on *Android* devices and PCs (tested on *Linux* but it should also be compatible with *Windows* and *m0acOS*), just like the *IPS Client*. It collects information about the capabilities of the devices in which it runs and submits it to the *YanuX Broker*, so that it can be used by the *YanuX Coordinator*'s distribution algorithm.

The *YanuX Orchestrator* allows to add to or override the collected information if something is missing or incorrect. It will also infer missing information. For instance, if the display's pixel density is known it can be used to approximate the *pixel ratio*[1], or vice

[1] Just like in *CSS* we define it as the ratio between physical pixels and logical pixels.

Table 3.1: Device Capabilities

Name	**Type/Units**	**Range/Examples**
Type	String	e.g., *desktop, laptop, tablet, smartphone* or *smart TV*
Display	**Single Object** or **Array**	
Type	String	e.g., *internal, external* or *touchscreen*
Orientation	String	e.g., *landscape* or *portrait*
Resolution	Array[Integer]/pixel	1 to ∞
Bit depth	Integer/bit	1 to ∞
Refresh Rate	Float/Hz	0 to ∞
Size	Float/millimeter	0 to ∞
Pixel Density	Integer/pixels per inch	0 to ∞
Pixel Ratio	Float	0 to ∞
Virtual Resolution	Array[Integer]/pixel	1 to ∞
Speakers	**Single Object** or **Array**	
Type	String	e.g., *loudspeaker* or *headphones*
Channels	Integer	1 to ∞
Bit depth	Integer/bit	1 to ∞
Sampling Rate	Float/Hz	0 to ∞
Camera	**Single Object** or **Array**	
Type	String	e.g., *main, telephoto, wide-angle, selfie* or *webcam*
Resolution	Array[Integer]/pixel	1 to ∞
Bit Depth	Integer/bit	1 to ∞
Refresh Rate	Float/Hz	0 to ∞
Microphone	**Single Object** or **Array**	
Channels	Integer	1 to ∞
Bit Depth	Integer/bit	1 to ∞
Sampling Rate	Float/Hz	0 to ∞
Input	Array[String]	*keyboard, mouse, stylus, touchscreen, speech input*
Sensors	Array[String]	*GPS, accelerometer, gyroscope, compass, barometer, light, proximity*

versa. Moreover, the *virtual resolution* of the display can also be calculated from the *pixel ratio* (Equations 3.1 and 3.2).

$$virtualResolutionWidth = resolutionWidth/pixelRatio \tag{3.1}$$

$$virtualResolutionHeight = resolutionHeight/pixelRatio \tag{3.2}$$

Developers can then just target the *virtual resolution* when they define their restrictions instead of the true physical resolution. Just like in *CSS*, working with these *virtual pixels* should be more consistent across different screens than working with physical pixels.

In the case of the development of the application prototypes presented in Chapter 5, our experimental setup often included a laptop running a *Linux* desktop operating system and a *Xiaomi Mi 8* smartphone running *Android 10* [GSM18]. Listings 3.1 and 3.2 show the gathered data about those devices as detected by the *YanuX Orchestrator*. Some of the information was manually added or edited, but additional information inference has yet to be performed. As mentioned in Chapter 6, we also performed tests with additional devices in preparation and during user studies. Table 6.1 shows a summary of the capabilities of the devices that were used.

Listing 3.1: *Linux* laptop capabilities

```
{"type":"laptop",
 "display":[{"type":"internal","resolution":[1920,1080],"bitDepth":24,
              "size":[345.5,196.5],"refreshRate":60},
            {"type":"external","resolution":[1920,1080],"bitDepth":24,
              "size":[477,268],"refreshRate":60}],
 "speakers":{"type":"loudspeaker","channels":2,"bitDepth":16,"
    samplingRate":44100},
 "camera":{"type":"webcam","resolution":[1280,720],"bitDepth":24,"
    refreshRate":60},
 "microphone":{"channels":1,"bitDepth":16,"samplingRate":44100},
 "input":["keyboard","mouse"],"sensors":[]}
```

Listing 3.2: *Xiaomi Mi 8* smartphone capabilities

```
{"type":"smartphone",
 "display":{"resolution":[1080,2248],"orientation":"portrait",
            "pixelDensity": 402,"bitDepth":24,"size":[69.5,142.5],
            "refreshRate":60},
 "speakers":{"type":"loudspeaker","channels":1,"bitDepth":16,
             "samplingRate":44100},
 "camera":[{"type":"main",
            "resolution":[4032,3024],"bitDepth":24,"refreshRate":60},
           {"type":"telephoto",
            "resolution":[4032,3024],"bitDepth":24,"refreshRate":60},
           {"type":"selfie",
            "resolution":[5184,3880],"bitDepth":24,"refreshRate":60}],
  "microphone":{"channels":1,"bitDepth":16,"samplingRate":44100},
  "input":["touchscreen","speech recognition"],
  "sensors":["gps","accelerometer","gyroscope","compass",
             "barometer","light","proximity"]}
```

In the future, we also intend to support collaboration and multitasking involving multiple users and devices within an ecosystem of *YanuX Framework*-enabled applications. We plan to have the *YanuX Orchestrator* listening for events coming from the *YanuX Broker* and react accordingly. For instance, it may launch or exit applications, split the screen for simultaneous multitasking in a public display, or deal with cross-device interaction gestures, such as dragging an item from one device to another and automatically start an app that deals with that item.

4

Indoor Positioning System for Pervasive Environments

This chapter is dedicated to present the development of the IPS solution which is required for the development of cross-device applications based on our framework that was previously mentioned in *3.2.4*. Moreover, this solution is one of the contributions of this thesis since it can also be used as a standalone system and integrated with other projects.

4.1 Architecture and Overview

A general overview of the architecture of the system is provided by Figure 4.1.

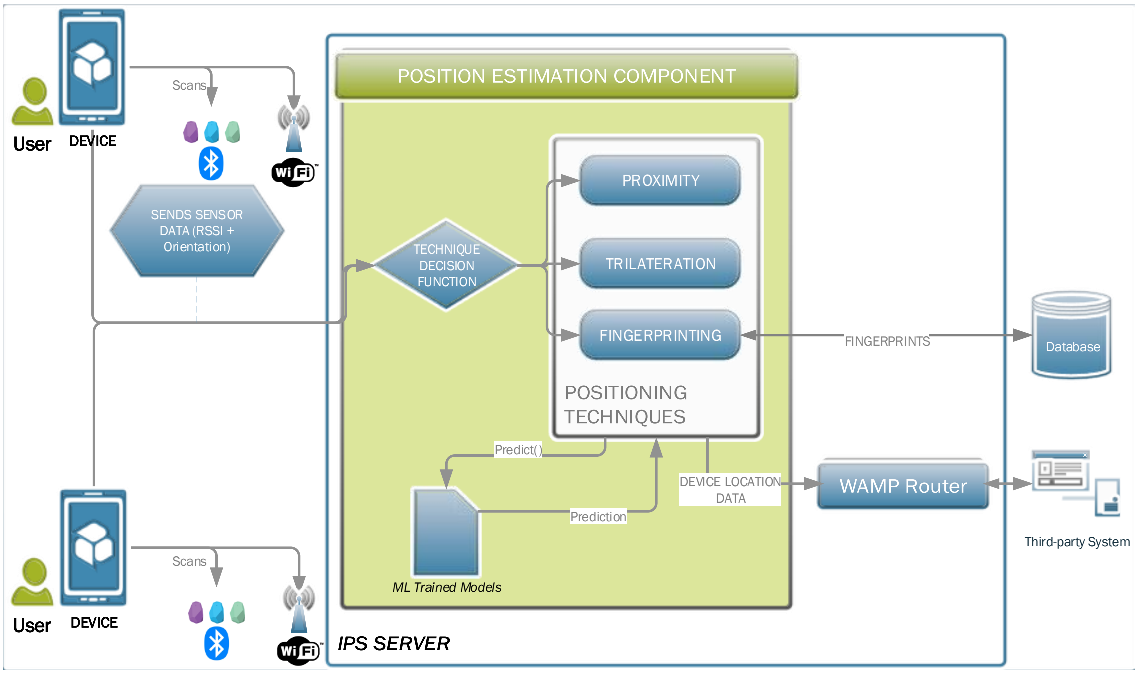


Figure 4.1: Architecture of the IPS system

The indoor positioning system was implemented by following a *client-server model*. The *client* runs on the devices which the system intends to locate. It has the responsibility of scanning for BLE beacons, *Wi-Fi* access points, and the orientation of the device, to send that information to the *server*. The *server* is responsible for determining the position of *clients* from that information using the supported positioning techniques. It is also capable to continuously communicating with the *clients* and third-party entities that are interested in the position of the *clients*.

The communication between *client* and *server* relies on a REST API to provide access to server resources. However, The communication between the *server* and third-party entities (e.g., the *YanuX Framework*) requires the ability to continuously push positioning events containing the current position of the client devices. Therefore, the WAMP protocol was used in this case since it provides an implementation of the Publish-Subscribe (PubSub) pattern over *WebSocket* connections [WAM21].

The *client* component has two implementations: the *IndoorApp* for mobile devices running *Android 6.0* or higher; and the *DesktopClient* for PCs running *Linux* (*Windows* and *macOS* are also theoretically supported but untested).

The *server* was implemented using the *Python* programming language, which allowed us to take advantage of many useful libraries for data analysis and machine learning. Since *Django* was used to build the REST API, we enjoy the flexibility of supporting multiple databases. During development we used *SQLite*, but *PostgreSQL* once the system was deployed.

BLE and *Wi-Fi* were the technologies that were integrated into the indoor positioning system. This decision mainly arises from the following set of benefits:

- **Significant ubiquity in modern devices**: both technologies have a widespread presence among today's modern devices, supporting the pervasiveness required for cross-device interactions in indoor settings.

- **Availability of hardware**: the substantial availability of *Wi-Fi* access points and the flexible deployment inherent to BLE beacons offers solutions with low cost and ease of deployment.

- **Performance**: the extensive research in indoor positioning systems that tackle *Bluetooth* and *Wi-Fi* technologies show that these technologies have the potential to perform relatively well in indoor positioning scenarios.

By taking the RSSI and orientation measurements, the system enables the application of two dimensions of the proxemic relationships: distance and orientation. The distance dimension is achieved in two different approaches:

- The prediction of the *proximity* technique is the distance between a device and a BLE beacon. In this sense, the distance dimension corresponds to the output of this positioning technique.

- The prediction of the *trilateration* and *fingerprinting* techniques is the micro-location or absolute location of a client device. The distance dimension can be inferred, i.e., by knowing the absolute location of two devices, it is possible to calculate the distance separating them, given that they are present in the same environment.

4.2 Scanning Procedure

The aforementioned *IndoorApp* and *DesktopClient* hold the responsibility of continuously scanning for BLE beacons, *Wi-Fi* access points, and the orientation of the devices. Additionally, they hold a UUID that identifies a device, allowing to match positions with known devices in a particular environment. Concerning the data saved in each sample, the RSSI values are saved for both BLE beacons and *Wi-Fi* access points, while the orientation value is represented by the *azimuth* in *radians* (i.e., rotation about the -*z-axis* as seen in Figure 4.2) [Goo21a; Goo21b].

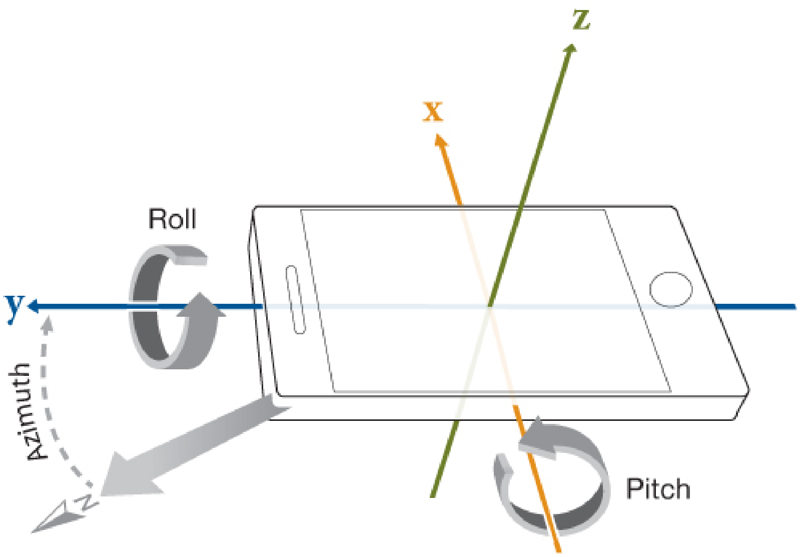


Figure 4.2: Coordinate system used by the rotation vector sensor on *Android* devices [Goo21a].

Previous research highlights a more severe fast fading multipath interference problem with BLE in indoor positioning settings than with *Wi-Fi* (mainly due to the shorter channel length in BLE) [FH15]. To alleviate such problem, besides saving a singular RSSI value for both technologies, the rolling average with a window size equal to the scanning rate (3 s) is also saved for BLE beacons. After every scan, the scanned samples are sent to the server.

The reason behind the decision of applying a scanning rate of 3 s comes from two different arguments:

- *Wi-Fi* has a slower broadcasting rate than BLE (i.e., a single scan can take multiple seconds) and only sends batched data (i.e., one report update gathers the RSSI from all the discovered access points) [FH14]. By setting the scanning rate for 3 s, although there is no guarantee of receiving an update from *Wi-Fi* in every scanning, there is very high chance that over the course of two consecutive scans (6 s) at least one *Wi-Fi* update will be received.

- According to Kim et al.'s work, the response time of a *proximity detection* positioning system, in the perspective of usability, given that humans are walking at a pace of 1.4 m/s on average, should range between 1 to 3 s [Kim+15].

4.3 Position Estimation

The *server* is responsible for estimating the position of the client devices through a set of positioning techniques: *fingerprinting*, *proximity* (distance estimation based on RSSI values), and *trilateration*. It also supports communication with third-party entities that may be interested in the positions of client devices.

The *server* uses *fuzzy logic* to choose fittest positioning technique for the current environment. In essence, it evaluates four variable features of the environment where the client device is integrated:

- Number of beacons scanned by the client device.

- Percentage (%) of access points matching between the access points scanned by the client's device and the access points featured in the supported radio maps, which have a substantial impact on the prediction (*feature importance*).

- Number of beacons matching between the beacons scanned by the client's device and the beacons featured in the supported radio maps, which have a substantial impact on the prediction (*feature importance*).

- Number of beacons scanned by the client device whose the server knows about their deployment coordinates in the current environment, i.e., it knows where the beacons are stationed inside an environment.

The inference procedure takes the inputs for each of the variable features and applies a set of rules to select one of the following positioning techniques as the one that best fits the environment:

- **Fingerprinting** takes advantage of machine learning algorithms to predict the outputs of unlabeled data. As a result, the fingerprints scanned in the *offline phase* are structured in a Comma-separated Values (CSV) file that denotes the radio map. The radio map is then trained through a machine learning algorithm before the server starts accepting the client's data. After that, the new data is solely inputted onto the trained radio map to predict the positions of devices.

- **Proximity** focuses on the notion of PBS, that is to say, the notion of relative location/context-based position (e.g., user *A* is near beacon *B*). Similarly to *fingerprinting*, *proximity detection* employs machine learning algorithms to predict the position of a device. However, although the system can function with *Wi-Fi* access points, it is best designed to work around BLE beacon signals since their deployment in the environment is more flexible and best suited to real-life scenarios.

- **Trilateration** uses the *proximity* technique to predict the distance separating the devices and multiple beacons. It then performs a least-squares optimization to predict the absolute location of client devices. As with *proximity*, *trilateration* is designed to work with both radio technologies, but BLE beacons are recommended.

Given the positioning technique resulting from the decision system, the positioning technique is applied to data received from the client device. When a prediction for a client device position is made, the *server* publishes the prediction to a specific topic on the WAMP router (*onLocationUpdate* in this case). This pattern allows any interested third-party entities to simply subscribe to the established topic to receive updates about the positions of client devices. In the case of the *YanuX Framework*, the position of devices in the environment is required to properly distribute UI elements. Therefore, the framework takes advantage of the position predictions through the previously presented *YanuX IPS Bridge* component which communicates with the IPS through this pattern.

4.4 Workflow

The workflow of the indoor positioning system when used in conjunction with the *YanuX Framework* is portrayed in Figure 4.3.

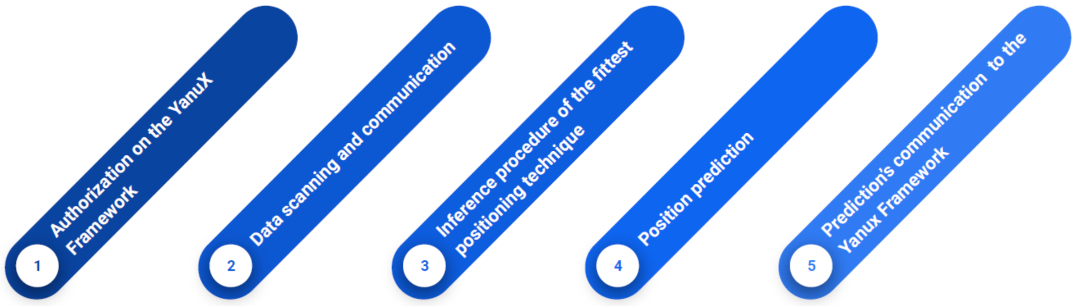

1
Authorization on the YanuX Framework
2
Data scanning and communication
3
Inference procedure of the fittest positioning technique
4
Position prediction
5
Prediction's communication to the Yanux Framework

Figure 4.3: IPS Workflow

The *clients* hold the execution of the *Authorization on the YanuX Framework* and *Data Scanning and Communication* stages. The *server* is responsible for the remaining steps.

4.4.1 Authorization on the YanuX Framework

This stage implemented is through the *client* applications, i.e., the *IndoorApp* for *Android* devices or the *DesktopClient* for PCs. It is responsible for processing the *OAuth 2.0* flow [D H12] and retrieving the device UUID. Regarding *OAuth 2.0*, the default authorization flow applied in the workflow is the *Authorization Code Flow with PKCE* [BA15].

This stage unfolds as follows:

1. The client application retrieves the client's device UUID by making a request to the *YanuX* platform.

2. If the client is not authorized yet (it does not have an *access token*), the application starts the *OAuth* authorization process. If the client is authorized, the validity of the access token is checked.
   If the token has expired, the refresh token is used to get a fresh access token.
   If the refresh token is invalid, clear all tokens and restart the process.

4.4.2 Data Scanning and Communication

In this stage, the *client* application takes over the execution flow. It focuses on scanning for *Wi-Fi* access points, BLE beacons, and the device's orientation. Therefore, this stage executes the following steps:

1. The client application sets up a periodic timer of 3 s, which represents the scanning rate.

2. During the scanning cycle, the application saves the RSSI values broadcast from the access points and the beacons, and the *azimuth* in *radians* (i.e., rotation about the *-z-axis*, as seen in Figure 4.2 [Goo21a; Goo21b]) from device's sensors.

3. Every 3 s, the client application sends a *POST* request to the server with the scanning updates retrieved during that time interval.

4.4.3 Inference Procedure of the Fittest Positioning Technique

This stage of the workflow harbors the decision system's execution based on *fuzzy logic*, designed to choose the positioning technique that best fits an environment. Consequently, its flow consists of the following execution steps:

1. Given the *crisp values* which serve as the inputs for the decision system, the server starts the decision system with the *fuzzification* process.

2. Through a set of predefined *rules* and the *fuzzy input sets*, the system infers each rule's strength.

3. With the aggregated rules' strength, the decision system outputs a *crisp value* corresponding to a positioning technique through a *defuzzification* process.

4.4.4 Position Prediction

This stage is responsible for predicting the position of a client device. It takes the positioning technique selected by the decision system and starts the following workflow:

1. Structures the client's device received data into formats compatible with the execution environments (e.g., CSV format);

2. Depending on the positioning technique, the server executes either a single prediction on a previously trained machine learning model for *fingerprinting* and *proximity*, or a combination of *proximity* with least-squares optimization for *trilateration*;

4.4.5 Prediction Communication to the YanuX Framework

As previously stated, the system supports communication with third-party applications, such as the *YanuX Framework*. Hence, this stage focuses on providing a continuous communication channel between the *IPS Server* and the *YanuX Framework*. The stage is structured as follows:

1. Structure the updated data in a supported format for both entities (*Server* and *YanuX Framework*);

2. Publish the newly structured data into the topic *onLocationUpdate*;

3. If *YanuX*, through the previously described *YanuX IPS Bridge* component, is subscribed to the topic, it will receive the new updates of the client devices' positions through a *WebSocket* connection following the WAMP protocol.

4.5 Positioning Techniques

The IPS supports multiple positioning techniques to support multiple environments and scenarios. The supported techniques are:

- *Fingerprinting*: An indoor technique that has enough accuracy for many scenarios and that can be implemented using multiple technologies. In our particular case we employ *Wi-Fi* and BLE.

- *Proximity*: A technique that focuses on detecting the presence and on estimating distance between devices. The IPS uses BLE to implement this technique. Although it only supports the relative position of devices, there are many scenarios in which relative position is more desirable than absolute positioning, including most of *YanuX Framework*'s use cases.

- *Trilateration*: A technique that can leverage the distance to several points to determine the absolute position of a device. In our system these distances are previously determined via the *proximity* technique.

The next subsections explain how each of the positioning techniques are implemented by *IPS Server*.

4.5.1 Fingerprinting

The *fingerprinting* technique consists of an *offline phase* responsible for data acquisition, and an *online phase* in which position predictions are made.

4.5.1.1 Offline Phase

The offline data acquisition phase of *fingerprinting* is started by the mobile application *IndoorApp*. The application is responsible for sending the fingerprint samples scanned in a particular environment to the *IPS Server*. Upon receiving this data, the server is responsible for completing the *offline phase* by constructing a radio map.

The server saves the fingerprint samples scanned by the *IndoorApp* application to a database. After completely scanning a given area (e.g., a room), the server will retrieve the samples from the database and convert it into a CSV file representing the radio map. Currently, this conversion is triggered by making a manual request to the */filter/* endpoint. Afterwards, the server terminates the *offline phase* by cleaning the fingerprints and associated data from the database.

4.5.1.2 Online Phase

The positioning of a client device using *fingerprinting* in a particular environment requires that a corresponding radio map is available. The *IPS Server* also needs to go through a training phase when it is initialized to use previously untrained radio maps. The output of this phase is a set of trained machine learning models, each associated with a dataset abstracting a radio map, capable of making predictions. Once the server is initialized, given the real-time scanned samples sent from the client devices through the *IndoorApp* or *DesktopClient* applications, and the inference of *fingerprinting* as the fittest positioning technique by the decision system, the server will use the appropriate pre-trained machine learning model to produce a position prediction based on the received scanned data.

4.5.2 Proximity

The *proximity* technique also consists of an *offline phase* responsible data acquisition, and an *online phase* in which position predictions are made.

4.5.2.1 Offline Phase

Instead of calculating distance from RSSI using a model based on path loss, the *IPS Server* uses machine learning for distance estimation. Therefore, an offline data acquisition phase is required to collect samples before real-time predictions can be made during an *online phase*. The *IndoorApp* is held responsible for data collection during this phase. It sends a *POST* request to the *IPS Server* with the scanned samples at specific distances from a reference beacon. Unlike *fingerprinting*, the server appends the scan results directly to a CSV file.

4.5.2.2 Online Phase

A radio map collected during the *offline phase* with RSSI values labeled at multiple distances is required in order to obtain a distance estimate using the *proximity* technique. The *IPS Server* also needs to go through a training phase when it is initialized if the *proximity* radio map has not been previously trained. The output of this phase is a machine learning model capable of making distance predictions based on RSSI values. Once the server is initialized, given the samples of an BLE beacon collected by a client device and that the decision system chose *proximity* as the fittest positioning technique, the server will prompt the pre-trained machine learning model for a distance prediction based on the received scanned data.

4.5.3 Trilateration

Trilateration differs from the other supported positioning techniques in that it does not require a dedicated *offline phase*. However, it relies on the *proximity* method to compute the distance between a client device and a set of beacons present in the surrounding environment. It also requires that a configuration file is present on the *IPS Server* with the known location of at least three beacons that are present in the environment.

Given the data samples sent from a client device and the choice of *trilateration* as the fittest positioning technique by the decision system, the server will execute the following steps:

1. Retrieve the locations of the captured beacons from the configuration file.

2. Compute the distance between the client device and the locations of the beacons using the *proximity* technique.

3. The estimation of the position of the device is achieved using the least-squares optimization to minimize the sum of squared errors.

The least-squares optimization process is required because in a real world scenario it is very unlikely that the circles defined by the calculated distances intersect each other on a single point. That situation leads to a set of equations without an analytical solution like in Figure 4.4.

Initially, the system could return a beacon configuration file corresponding to the wrong room, as long as a device was still capable of detecting enough beacons from an adjacent room, or that were otherwise purposely shared between two different environments. Therefore, this method had to be improved to properly support this scenario.

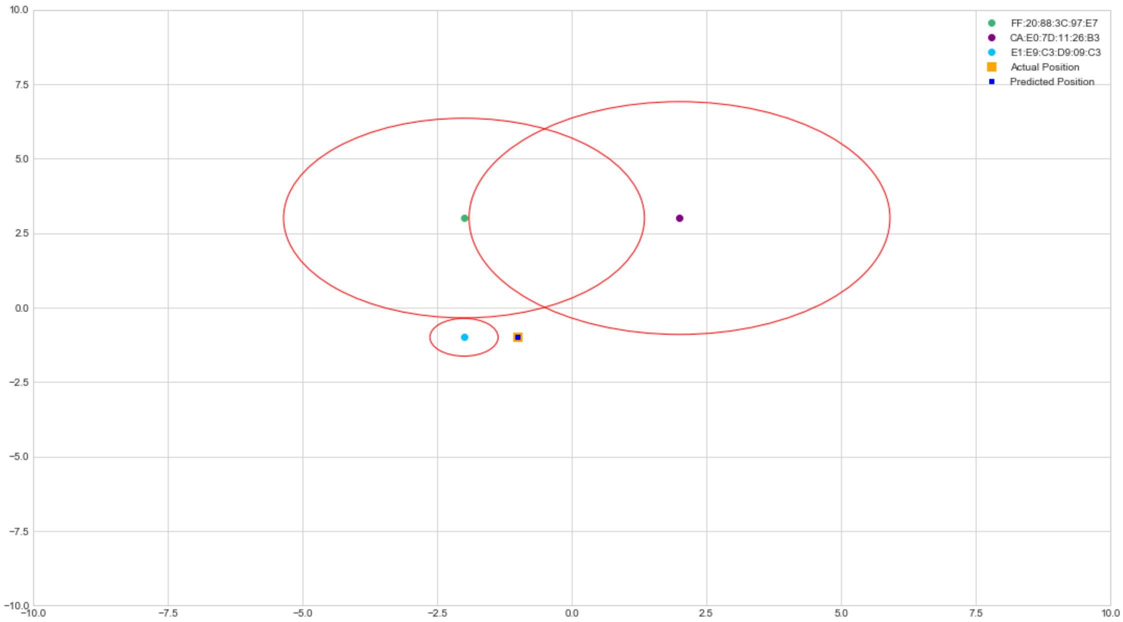


Figure 4.4: *Trilateration* – Least-squares Optimization

The solution was to evaluate the RSSI of the beacons listed in the configuration file corresponding to each place and select the one which one had the largest average value. This decision assumes that larger RSSI values translate to closer beacons, which indicates that the a given site is more likely to be the one where the device is present. From our empirical tests, this largely eliminated the issue.

4.6 Fuzzy System

Since the IPS supports multiple positioning techniques, it is important to select the most suitable one for each situation. Therefore, the system also contains a decision component that infers the positioning technique that best fits the environment the device is in. This decision component is built on *fuzzy logic* by drawing inspiration from previous work by Orujov et al., which also uses *fuzzy logic* to decide the positioning technique that best fits an environment according to a set of characteristics of the environment (room size, RSSI, and quantity of beacons) with the objective of achieving an accuracy at least equal to the best performing technique [Oru+18].

Fuzzy logic is a generalization of *crisp logic*, in which logic operations (*AND*, *OR*, and *NOT*) are restrained to output either an entirely true value (degree of truth of 1) or a completely false value (degree of truth of 0). However, in *fuzzy logic*, the degree of truth of a concept can have a value ranging between 0 and 1 [Ame]. Hence, unlike standard boolean logic applied in computers, this type of logic resembles human decision-making, which often has inherently vagueness and uncertainty in the set of possible decisions [Tut].

This resemblance may lead to contexts where the output is not entirely true but partially true (e.g., degree of truth of 0.7). This means that *fuzzy logic* is suitable for scenarios with a certain degree of uncertainty, as is the case of indoor settings where the environment variability translates to uncertainty in choosing the best positioning technique to apply.

Moreover, a *fuzzy logic* system applies *fuzzy set* theory (*fuzzy sets* and rules describing the system behavior) to map inputs (features in *fuzzy classification*) to outputs (classes in *fuzzy classification*) based on an inference knowledge-based approach [Pri]. The heart of a *fuzzy system* is a knowledge base consisting of the so-called *fuzzy IF-THEN* rules [Wan97].

There are several types of *fuzzy inference systems* such as *Mamdani*, *Sugeno* or *Tsukamoto* systems. In our case we implemented a decision system based on the *Mamdani* model as part of our IPS solution [MA75]. It is structured in the following stages (see Figure 4.5):

- **Fuzzification**: Mapping of *crisp input* parameters into *fuzzy sets* quantified by a membership function. Basically, it translates input values into linguistic variables.

- **Rule Base**: Determination of the set of *if-else* conditions (i.e., rules) that form linguistic information based on *fuzzy variables*.

- **Inference Procedure**: Focuses on simulating human reasoning by applying inference on the fuzzy input sets and the rules to determine the rule strength. This stage ends with the clipping of the output membership function at the rule strength and the following aggregation of the set of the consequences of the *fuzzy rules* (clipped membership functions) that provides a singular *fuzzy output* distribution.

- **Defuzzification**: This stage applies a *defuzzification* technique to obtain a *crisp output* value from the *fuzzy output* distribution.

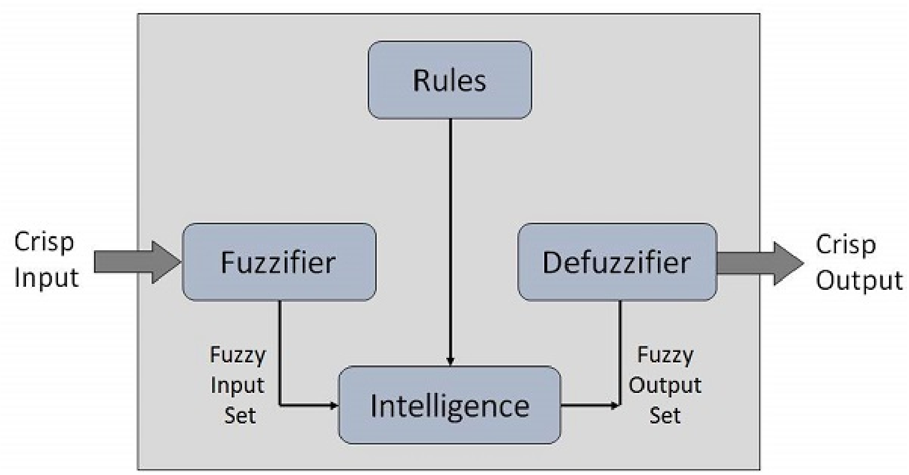


Figure 4.5: *Fuzzy Logic* Architecture [Tut]

The subsequent subsections present each stage of the *fuzzy system* in further depth.

4.6.1 Fuzzification

In this phase, the *crisp values* inputted into the decision system are mapped to values between 0 and 1 using a set of membership functions.

4.6.1.1 Crisp Inputs

The following *crisp input* parameters were considered:

- **Number of Beacons**: Number of BLE beacons in the environment.

- **Matching Access Points**: Matching percentage of access points captured by the client device whose weight (feature importance) in a radio map surpasses a certain threshold (0.005).

- **Number of Matching Beacons**: Number of matching of beacons captured by the client device whose weight (feature importance) in a radio map surpasses a certain threshold (0.005).

- **Beacons Location**: Number of beacons captured by the client device whose location is known by the system.

Concerning the *Matching Access Points* parameter, the initial decision was to analyze the matching percentage of the captured access points in the existing radio maps. If there was a matching percentage of at least 50%, then the radio map would be targeted for *fingerprinting* if the decision system selected this positioning technique. However, such a proposition did not recognize the weight of the access points as features in a machine learning algorithm. That could lead to selecting a radio map whose captured access points had no relevance for the computation of the training and prediction phases. Therefore, the proposed system finds the matching percentage of captured access points whose relevance (feature importance) in the radio map surpass the threshold of 0.005. This value was chosen by analyzing bar charts of feature importance in several radio maps and concluding that this value would not cut out too much the set of relevant access points while filtering the almost meaningless access points.

Moreover, the decision to have a different metric method for BLE (number of matching beacons) and *Wi-Fi* (percentage of matching access points) is explained by the uncertainty on the number of access points that provide efficient performance. With BLE, *fingerprinting* accuracy increases with up to 10 deployed beacons according to studies by Faragher et al. and Martins et al [FH15; Mar+20]. Nevertheless, Orujov et al. conclude that 3 beacons can be enough to maintain satisfactory performance across the three positioning techniques [Oru+18]. Therefore, the system evaluates a discrete parameter for BLE and a continuous value for *Wi-Fi*.

4.6.1.2 Membership Functions

The *fuzzy logic* process maps *crisp input* values into values in the interval between 0 and 1 by finding the intersection of these values with the associated input membership functions. A membership function quantifies linguistic variables by computing the degree of membership (a value between 0 and 1) of an input. Therefore, there is an associated membership function for each input parameter.

4.6.1.2.1 Number of Beacons

Figure 4.6 displays the triangular membership function associated with this input parameter corresponding to the following linguistic variables:

- *None*: 0 beacons captured.
- *Medium*: 1 to 2 beacons captured.
- *Good*: 3 or more beacons captured).

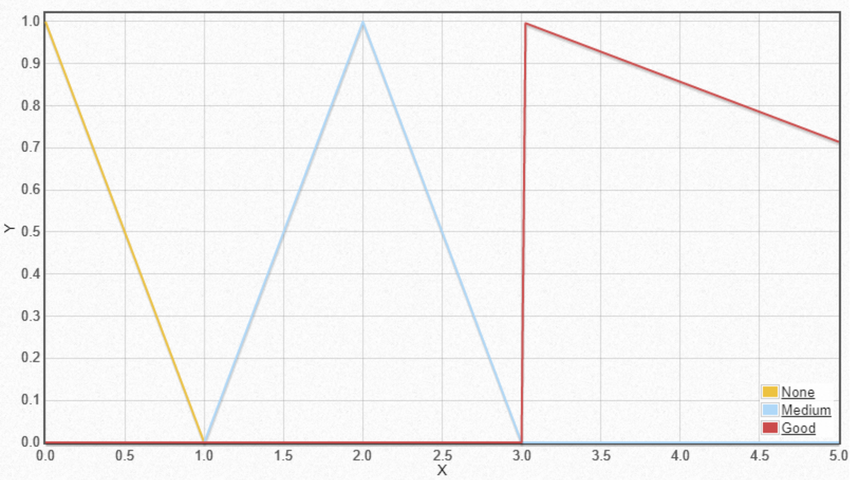


Figure 4.6: Membership Function – Number of Beacons

4.6.1.2.2 Matching Access Points

This input parameter is associated with a trapezoidal membership function (see Figure 4.7) corresponding to the following linguistic variables:

- *Not Enough*: 35% matching or less of the access points with relevancy.
- *Enough*: More than 35% matching of the access points with relevancy.

If the 35% threshold was not applied, then it would be cumbersome to find a set of captured access points in which all were relevant for the estimation of predictions.

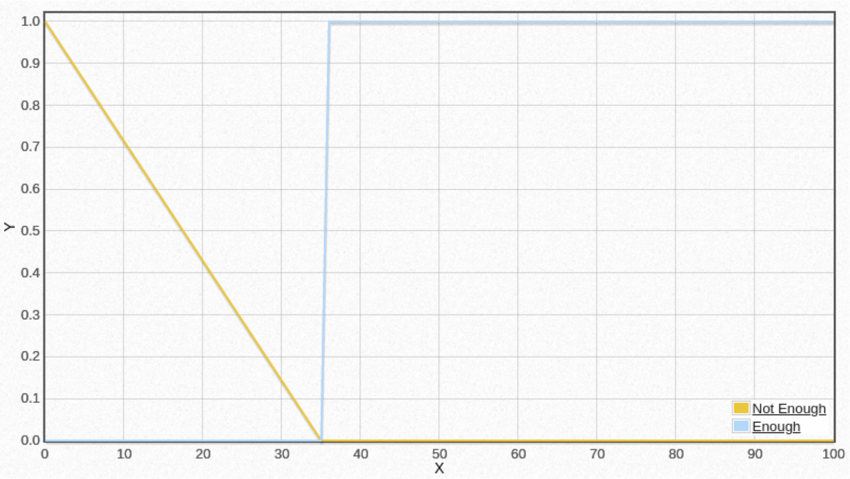


Figure 4.7: Membership Function – Number of Matching Access Points

4.6.1.2.3 Number of Matching Beacons

Figure 4.8 shows the trapezoidal membership function supporting this input parameter corresponding to the following linguistic variables:

- *Not Enough*: Less than 3 matching beacons.
- *Enough*: 3 or more matching beacons.

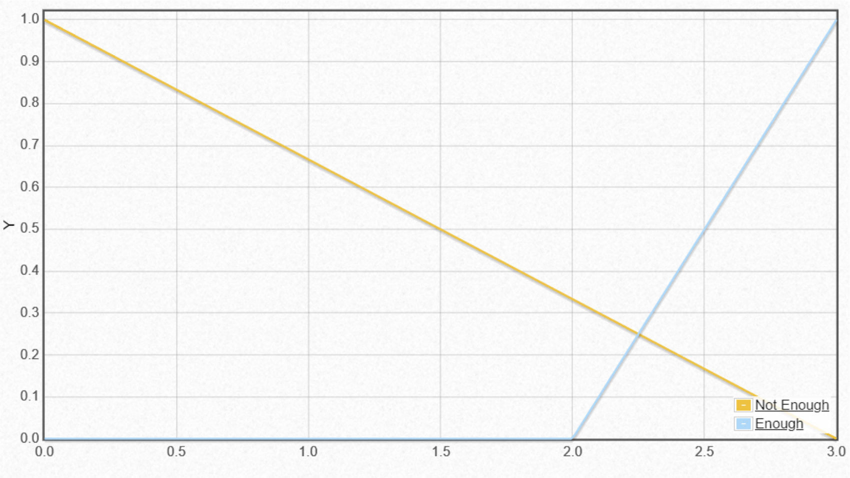


Figure 4.8: Membership Function – Number of Matching Beacons

4.6.1.2.4 Beacons Location

This input parameter allows to employ the *trilateration* positioning technique since it holds the logic behind the position of the deployed beacons in the environment. Figure 4.9 displays the trapezoidal membership function associated with this input which corresponds to the following linguistic variables:

- *Not Available*: Less than 3 known positions of beacons);
- *Available*: 3 or more known positions of beacons).

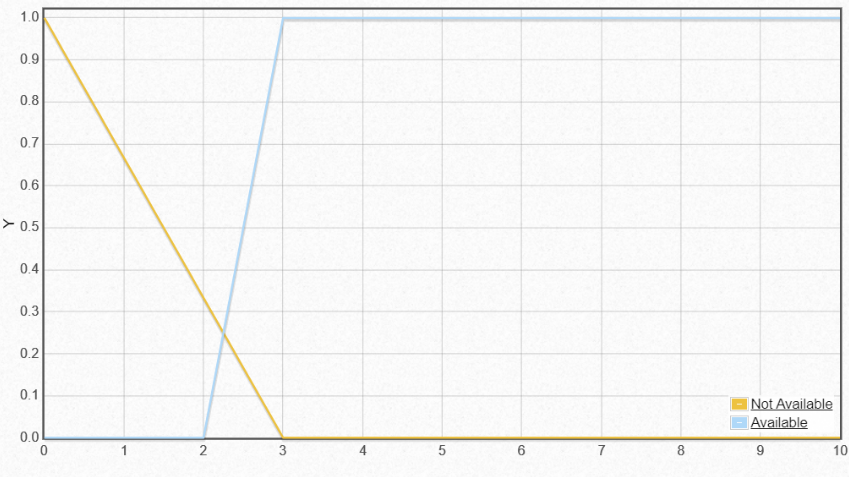


Figure 4.9: Membership Function – Number of Known Beacons Locations

4.6.1.2.5 Output

The output value is also supported by a membership function. Figure 4.10 depicts a Gaussian membership function ($spread = 0.1$) with the linguistic variables being the positioning techniques:

- *Proximity*: $mean = 0.5$;
- *Trilateration*: $mean = 1.5$;
- *Fingerprinting*: $mean = 2.5$.

4.6.2 Rule Base

This component contains the set of *if-else* rules based on linguistic information. These rules are leveraged in the inference procedure, along with the fuzzified inputs. The decision system's rules support the performance results by establishing a hierarchy among the available positioning techniques according to their overall performance in the experiments described in section 4.8.

The rules applied by the decision component are:

1. If **Number of Beacons** is *None* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Enough* then **Position Technique** is *Fingerprinting*.

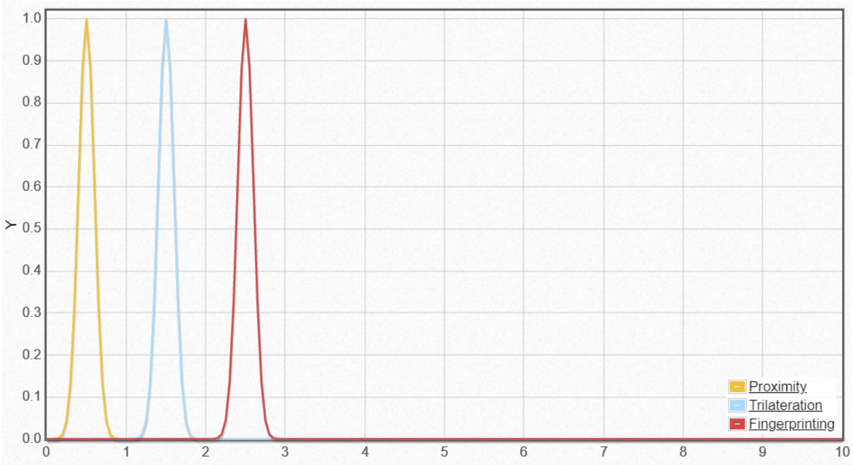


Figure 4.10: Membership Function – Output Positioning Technique

2. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Enough* and **Matching Access Points** is *Enough* and **Beacons Location** is *Not Available* then **Position Technique** is *Fingerprinting*.

3. If **Number of Beacons** is *Medium* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Enough* and **Beacons Location** is *Not Available* then **Position Technique** is *Fingerprinting*.

4. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Enough* and **Matching Access Points** is *Not Enough* and **Beacons Location** is *Not Available* then **Position Technique** is *Fingerprinting*.

5. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Not Enough* and **Beacons Location** is *Available* then **Position Technique** is *Trilateration*.

6. If **Number of Beacons** is *Medium* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Not Enough* then **Position Technique** is *Proximity*.

7. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Not Enough* and **Beacons Location** is *Not Available* then **Position Technique** is *Proximity*.

8. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Enough* and **Matching Access Points** is *Not Enough* and **Beacons Location** is *Available* then **Position Technique** is *Trilateration*.

9. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Enough* and **Matching Access Points** is *Enough* and **Beacons Location** is *Available* then **Position Technique** is *Trilateration*.

10. If **Number of Beacons** is *Good* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Enough* and **Beacons Location** is *Available* then **Position Technique** is *Trilateration*.

11. If **Number of Beacons** is *None* and **Number of Matching Beacons** is *Not Enough* and **Matching Access Points** is *Not Enough* then **Position Technique** is *None*.

The rules outline a hierarchy of the positioning techniques with *trilateration* being the default technique that is used when all conditions are *good enough*. Therefore, the hierarchy is *trilateration* > *fingerprinting* > *proximity*. Nonetheless, the prioritization of *fingerprinting* above *proximity* is not so much about performance results but about the information provided. *Fingerprinting* provides the absolute position of a device allowing to compute the relative position in relation to another absolute position. However, *proximity* only provides displays the relative position in relation to a beacon in the environment, a beacon attached to a device, or a beacon signal emitted by a device.

4.6.3 Inference Procedure

The fuzzy inference module is structured in three stages:

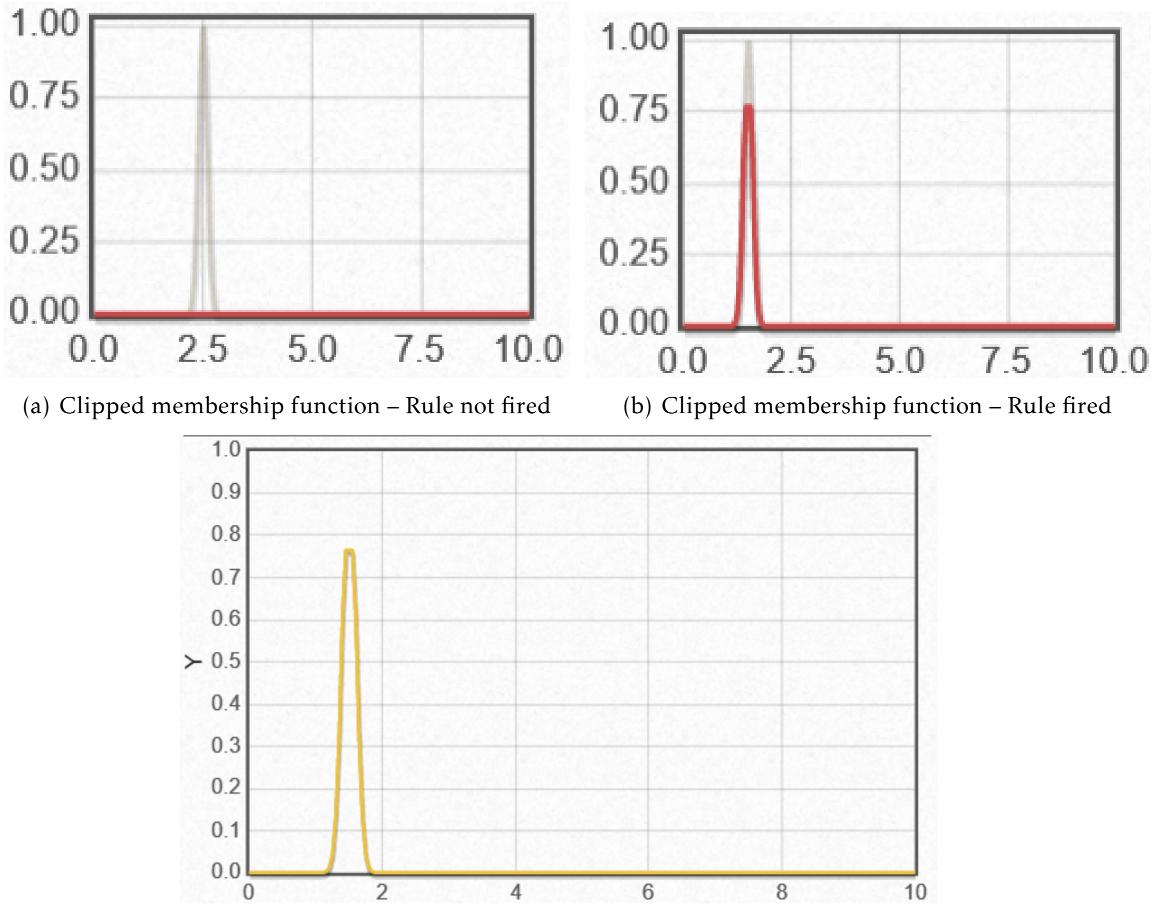
1. **Computation of Rule's Strength**: Application of fuzzy operator *AND* in the combination of the fuzzified input parameters.

2. **Clipping the Output Membership Function**: This stage fires rules according to the clipping of the output membership function at each rule's strength. Figure 4.11(a) and 4.11(b) displays both scenarios in which rules are not fired and fired, respectively.

3. **Aggregation**: The fuzzy operator *OR* is used to aggregate the clipped membership functions (rule's consequence). The final result of this stage is the fuzzy output distribution function (see Figure 4.11(c)).

4.6.4 Defuzzification

Since the decision system's desired output is a crisp string representing the chosen positioning technique, the system must process the fuzzy output distribution function through a *defuzzification* technique to retrieve a crisp value. The system uses the centroid technique to narrow the system to one crisp value by finding the center of mass of the output distribution. This crisp number is then rounded to determine the equivalent positioning technique.



(a) Clipped membership function – Rule not fired

(b) Clipped membership function – Rule fired

(c) Output Distribution Function

Figure 4.11: Inference Procedure

4.7 Client Application

IndoorApp is the *Android* implementation of the *IPS Client*. It enables *Android* devices to be tracked by the indoor positioning system. Along with the *DesktopClient*, it has a vital role in the lifecycle of the entire system since it takes on the entity of the *client* at the system level.

The *DesktopClient* is a more recent development given the need to support traditional PCs during user studies. As it stands, it is simple Command Line Interface (CLI) application implemented in *Node.js* that only supports the most basic features of user authorization and scanning. This section focuses mostly on detailing the implementation of the *IndoorApp*, which was the featureful initial reference implementation of a *client application* for the indoor positioning system. These are the main features provided by *IndoorApp*:

- **Scanning Procedure**: Scanning for *Wi-Fi* access points, BLE beacons, and the device orientation is the main purpose of the mobile application.

- **OAuth 2.0 Authorization**: *IndoorApp* supports *OAuth 2.0* client authorization to access data in third-party systems like the *YanuX Framework*.
- **Positioning Techniques Procedures**: Even though the core responsibility of the application is providing scans to update the device's position in real-time, it also supports the *offline phase* of the *fingerprinting* and *proximity* techniques. The application allows the scan of data at reference points in the environment and provides a testing tool to assess the prediction quality in real-time for all the positioning techniques.

When a user opens the mobile application, the main screen displays the following options (see Figure 4.12):

- **Register**: By default, it allows a user to register an account with the *YanuX Auth*. It can be reconfigured to support another authentication/authorization backend.
- **Login**: Allows a user to authenticate in the *YanuX Auth*. It can be reconfigured to support another *OAuth 2.0 Authentication Server*. If the client has not been yet authorized to access a user's resources, this flow will automatically start the authorization flow. Otherwise, it will display a toast to the user with the message *Client already authorized*.
- **Experimental**: This button gives access to the aforementioned positioning technique procedures.
- **Scan**: Initiates the procedure which continuously scans for sensor data in the background;

4.7.1 Authorization

The *IndoorApp* supports *OAuth 2.0* as a mechanism for client authorization in third-party systems. Figure 4.13 portrays a general overview of the *IndoorApp* authorization as a client in the *YanuX Framework*. The *DesktopClient* follows a similar approach. In both cases, the applications can be reconfigured to integrate with other authentication providers following the same standards. Furthermore, the applications support the *Authorization Code* and *Authorization Code with PKCE* flows. *PKCE* is an extension of the standard *Authorization Code* flow that provides additional security mechanisms and it is the one that is used by default.

The standard workflow of the authorization process of the client applications follows these steps:

1. If the access token exists, the service checks its validity (expiration date);

    a) If the access token is valid, the application can proceed.

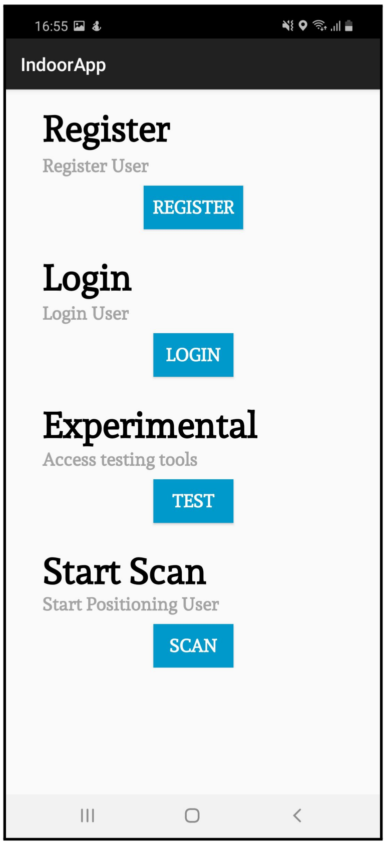


Figure 4.12: *IndoorApp* – Main Screen

b) If the access token is invalid, the service makes request to acquire a new access token using the refresh token.

c) The flow ends with a call to a *Token Introspection* endpoint which returns meta-information about the token and the resource owner as defined in *RFC 7662* [Ric15];

2. If the access token does not exist, the service will follow the *PKCE* flow to acquire an access token and a refresh token.

Besides going through the authorization process, the client applications must also make a request to a local endpoint that exposes the UUID of the device where the application is running. This UUID is generated the YanuX Orchestrator why by default runs an endpoint at `http://locahost:3003/deviceInfo` that returns a JSON object with the a property called *deviceUuid* containing. This is something that other systems that intend to integrate with our IPS solution must also provide in order to identify the devices. Alternatively, the IPS could be easily modified to be the one generating UUID for third-party systems.

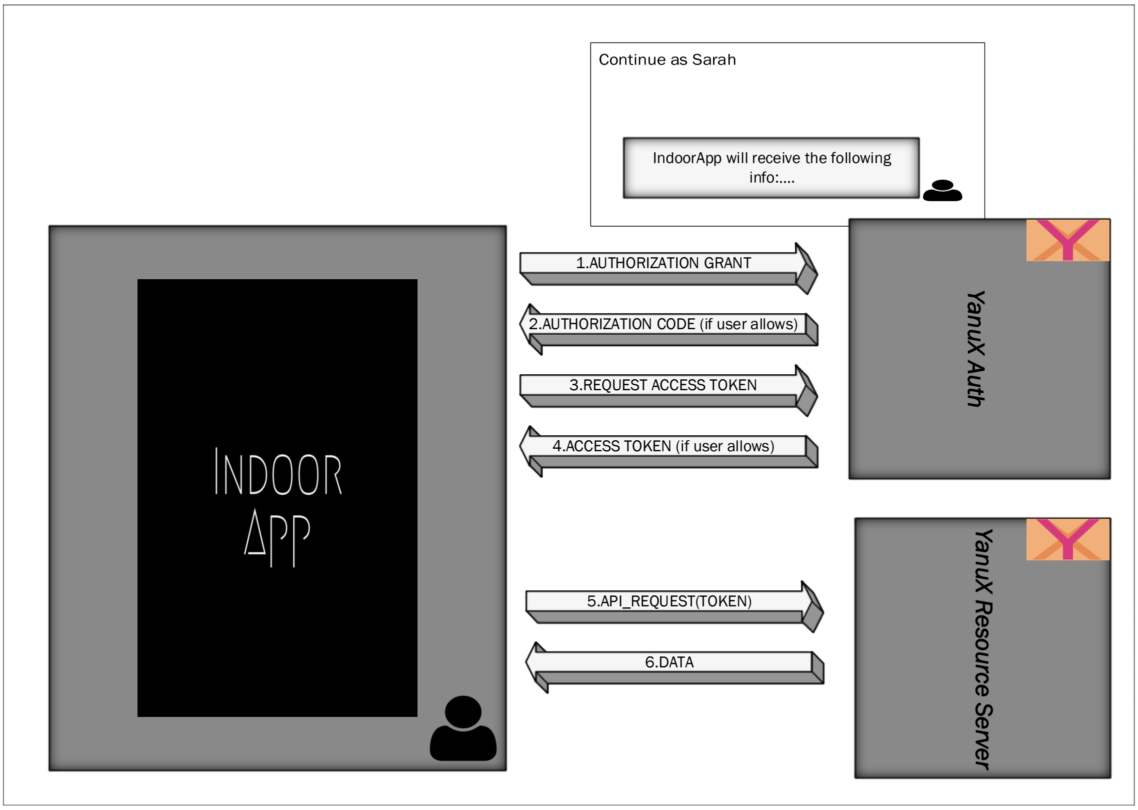
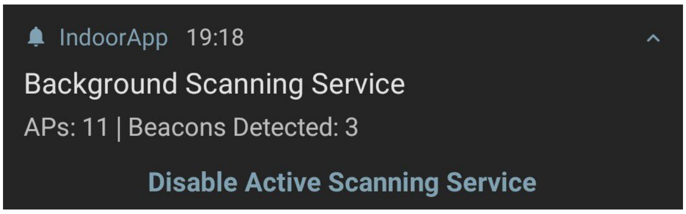


Figure 4.13: *IndoorApp* – OAuth 2.0 Authorization

4.7.2 Scanning Procedure

Once the user clicks on the *Scan* button on the main screen of the *IndoorApp* (Figure 4.12), the application starts a service responsible for scanning data for *Wi-Fi* access points, BLE beacons, and for capturing the orientation of the device. The service displays a persistent notification to the user with the following information (see Figure 4.14).



IndoorApp 19:18

Background Scanning Service

APs: 11 | Beacons Detected: 3

Figure 4.14: *IndoorApp* – Scanning Notification

Every 3 seconds, the service starts scanning for *Wi-Fi* access points, BLE beacons and changes in the orientation sensor. The service continuously gathers information data during the scanning period. The data collected about *Wi-Fi* and BLE, they share some similarities, e.g., RSSI values., but they have distinct capturing flows and sample structures based on the APIs used to collect them. Once the time elapses, the service sends a *POST* request to the *IPS Server* with the collected data. The *DesktopClient* application implements the same approach as part of its scanning procedure.

4.7.2.1 Bluetooth Low Energy (BLE)

The *IndoorApp* scans *iBeacon* packets [App18] using the *Android Beacon Library* [Rad]. This library enables the configuration of the following parameters:

- **Duration of each BLE scan cycle**: This value was set to 150 ms.
- **Time between BLE scanning cycles**: The value was configured to 0 ms since the service wants continuously scan for beacons.

The *DesktopClient* implementation of the scanning procedure was developed to mirror this behavior as closely as possible using the *noble* library for *Node.js* [aba21].

We used beacons from an *Estimote Developer Preview Kit* [Est18] that were configured to emit *iBeacon* packets [App18].

Figure 4.15: An *Estimote Developer Preview Kit* containing 3 beacons.

The *Estimote* beacons were configured as follows:

- **Transmit Power**: −12 dBm;
- **Advertising Interval**: 100 ms;

Under certain circumstances we also configured mobile devices to broadcast their own *iBeacon* packets instead of relying on an external beacons to identify them. We tried to match the settings we used for the beacons as closely as possible.

Faragher and Harle's work justifies the reasoning behind the of configuration of the *Estimote* beacons and *Android Beacon Library* [FH15]. According to their studies, the transmit power of beacons should range between −10 to −20 dBm. Therefore, we used the value −12 dBm that is set by default on *Estimote* beacons. Moreover, increasing the transmit power would negatively impact battery life.

The authors also analyzed the impact of the beacon advertising rate. They observed a decrease in the positioning errors with increasingly faster advertising rates at a cost of decreasing the battery life of the beacons. We settled with an advertising rate of 10 Hz (100 ms). However, we may eventually decrease this rate to optimize battery usage. The *Android Beacon Library* scanning cycle is set to a slower rate of 6.5 Hz (150 ms) to reduce the likelihood of not getting a sample from each beacon during each BLE scan period.

Regarding the data structure used to save received BLE advertisement samples, the application sends to the server an array of objects, each containing the following variables:

- **Media Access Control (MAC) Address**: MAC address of a beacon.
- **iBeacon Parameters**: UUID, major, and minor values of a beacon.
- **Current RSSI Value**: Last known RSSI value of a beacon.
- **RSSI Values**: An array holding the RSSI values scanned during the scanning window (3 s). This is used to calculate a rolling mean to alleviate multipath interference problems.

4.7.2.2 Wi-Fi

The configuration of *Wi-Fi* access points is not easily accessible. This means that the system to simply scans the environment without control over the configuration of the signals being emitted. Moreover, unlike *BLE*, scan results for access points are reported in a single batch report after an uncertain amount of time [FH15]. Therefore, the scanning service may not have an updated report on every scanning cycle (3 s). When that happens, it will have to reuse *Wi-Fi* scanning results from a previous cycle.

The data structure sent to the server concerning *Wi-Fi* consists of an array of objects, each containing the following information about an access point:

- **MAC Address**: The MAC address of an access point.
- **RSSI Value**: This last known RSSI value of an access point.

The RSSI-based rolling average is not used with *Wi-Fi* because it would be unpractical given how long each scan takes to complete. By the time enough data was gathered it would be too out-of-date to be of any use, e.g., the user could have already moved several meters. Besides that, *Wi-Fi* channels are not as prone to multipath interference as BLE since they significantly wider (20 MHz).

4.7.2.3 Orientation

The orientation sensor provides relevant information about the direction that devices are facing. In the case of the *YanuX Framework*, this is used to establish proxemic relationships according to the orientation of the devices. These relationships are then used to decide how to automatically distribute the UI elements across devices.

In the *IndoorApp*, the orientation of a device can be extracted from built-in sensors by following these steps:

1. Listen for changes in the magnetometer and accelerometer sensors.

2. When the data changes in both sensors, compute a *rotation matrix* that allows the extraction of three orientation angles (*azimuth*, *pitch*, and *roll* as seen in Figure 4.2).

3. Adjust these values taking into account the orientation of the device (portrait or landscape).

4. The orientation that is reported by the system will be the *azimuth* parameter. This value represents the angle of rotation about the *-z axis*. It represents the angle between the device's *y axis* and the magnetic north pole. The value varies between $-\pi$ to π and can be interpreted like a compass: *North* is 0, *South* is π, *East* is $\pi/2$, and *West* is $-\pi/2$.

The *DesktopClient* is meant to run on a laptop or desktop PCs. This devices rarely have the sensors required to determine their own orientation. Therefore the orientation must be determined by another device. It can then be specified as part of a configuration file before launching the *DesktopClient*.

4.7.2.4 Reporting to the IPS Server

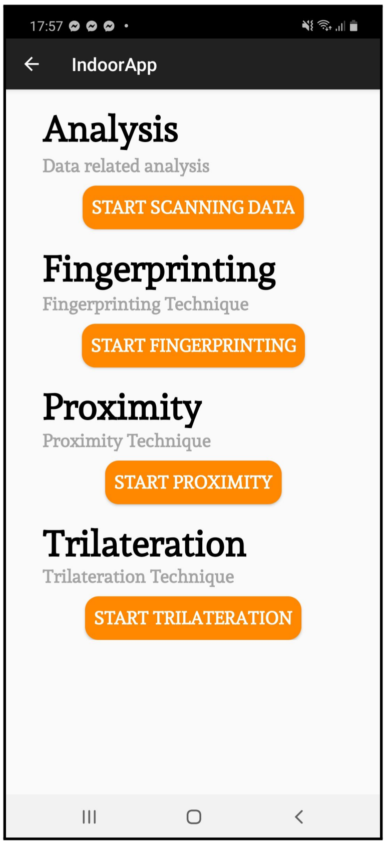
At the end of the 3 s scanning cycle, the service holds information about the *Wi-Fi* access points, BLE beacons and the orientation of the device. The service will end the execution flow with a *POST* request to the *IPS Server* at the */scanning* endpoint. The data structures holding the gathered sensor data are sent in the body of the request after being converted to *JSON*. Thereby, Table 4.1 shows an overview of the data communicated from the *IndoorApp* or *DesktopClient* to the *IPS Server*.

Table 4.1: Communication between *IndoorApp* and *IPS* server

Technology	**Sent Structure**
Wi-Fi	Current RSSI + MAC Address
Bluetooth	Current RSSI + RSSI Rolling Average + MAC Address + *iBeacon* Parameters
Orientation	*Azimuth* Angle

4.7.3 Additional Tools Provided by the *IndoorApp Android* Application

The main responsibility of the *IndoorApp* is to scan and communicate sensor data to the *IPS Server*. However, it is also responsible for collecting data during an *offline phase* to train the models used by the server to make the predictions required by *fingerprinting* and *proximity* techniques. The application also allows to test the predictions made during the *online phase* by showing them directly on the application. These tools can be accessed through the *Test* button on the main screen of the application (see Figure 4.12) which gives access to the screen displayed in Figure 4.16.



Analysis
Data related analysis

Fingerprinting
Fingerprinting Technique

Proximity
Proximity Technique

Trilateration
Trilateration Technique

Figure 4.16: *IndoorApp* – Experiment Screen

This screen presents the following options:

- **Analysis**: Shows the sensor data being captured by the application in real-time.
- **Fingerprinting**: Gives access to the *fingerprinting* tools.
- **Proximity**: Gives access to the *proximity* tools.
- **Trilateration**: Gives access to the *trilateration* tools.

4.7.3.1 Fingeprinting

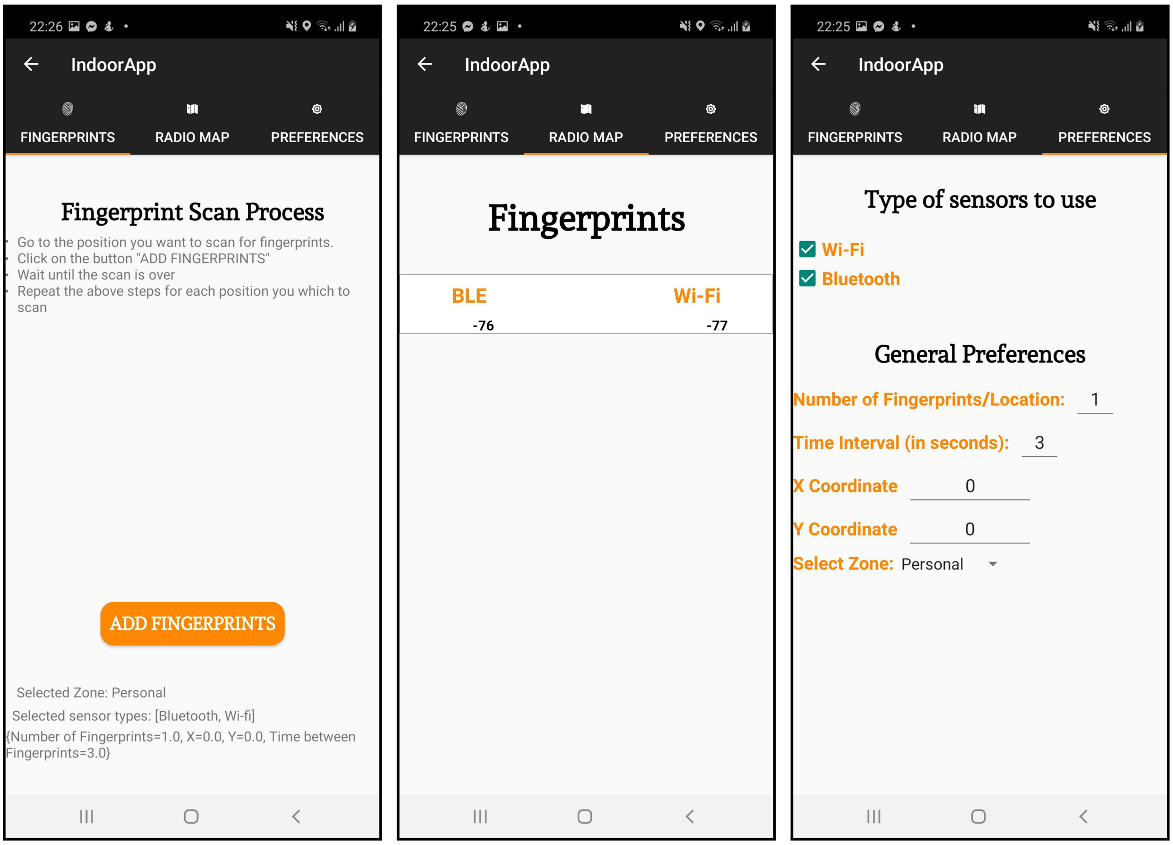
Fingerprinting is comprised of an *offline phase* to capture fingerprints at reference points and an *online phase* to predict the location of the devices by a matching data captured in real-time with the fingerprints associated with the surrounding environment. The *IndoorApp* implements both phases and provides a tool to test the *online phase*.

For the *offline phase* it offers the following options (see Figure 4.17):

- **Main Screen** (Figure 4.17(a)): A device can start adding fingerprints to a radio map in a particular environment by clicking on *Add Fingerprints*. This screen also displays instructions about the fingerprint scanning process and the current settings for the fingerprint scanning cycles.

- **Radio Map** (Figure 4.17(b)): This screen shows the fingerprints that have already been saved.

- **Preferences** (Figure 4.17(c)): This screen displays the configuration parameters supported by the *offline phase*.



Fingerprint Scan Process

- Go to the position you want to scan for fingerprints.
- Click on the button "ADD FINGERPRINTS"
- Wait until the scan is over
- Repeat the above steps for each position you which to scan

Selected Zone: Personal

Selected sensor types: [Bluetooth, Wi-fi]

{Number of Fingerprints=1.0, X=0.0, Y=0.0, Time between Fingerprints=3.0}

Fingerprints

BLE	Wi-Fi
-76	-77

Type of sensors to use

☑ Wi-Fi
☑ Bluetooth

General Preferences

Number of Fingerprints/Location: 1
Time Interval (in seconds): 3
X Coordinate 0
Y Coordinate 0
Select Zone: Personal

(a) Main Screen (b) Radio Map Screen (c) Preferences Screen

Figure 4.17: *IndoorApp* – *Fingerprinting* Offline Screens

By default, the *offline phase* scanning cycle was set to 10 s in order to collect enough data per fingerprint to be used by the machine learning algorithms. Moreover, the user performing the *offline phase* must have some training about the system beforehand. Upon the end of a scanning cycle, the application will send *POST* requests with the gathered data to endpoints provided by the *IPS Server*. The sent data includes:

- List of *Wi-Fi* access points and BLE beacons captured, each being an object holding the MAC address and the RSSI value;

- The absolute position of the reference point in which the fingerprint was scanned;

The *IndoorApp* also provides a real-time testing tool of the predictions executed in the *online phase*. Figure 4.18() shows the prediction visualization screen, and Figure 4.18(a) shows the configuration screen with the following options:

- **Machine Learning Algorithm**: Selection of the algorithm to use to predict the device's position.

- **Filter**: Select if a filter (median or mean) should be applied to the radio map's fingerprints, i.e., fingerprints will be replaced by the median/mean of RSSI values of access points on each reference point.

- **Technologies Used**: Selection of the technologies used to predict the device's position. This parameter should be consistent with the technologies applied in the *offline phase*.

As with the *offline phase*, the application uses a scanning rate of 10 s. It will then send a request to an endpoint provided by the *IPS Server*.

4.7.3.2 Proximity

The *proximity* technique predicts the relative position of devices (e.g., the device is 5 m away from the beacon *B*, or the device is in the *Personal* zone of beacon *C*). The system works similarly to *fingerprinting* by having an *offline phase* and an *online phase*.

The application scans for *iBeacon* packets while standing at specific distances from a reference *BLE* beacon during the *offline phase*. Figure 4.19 shows the screens that enable this process:

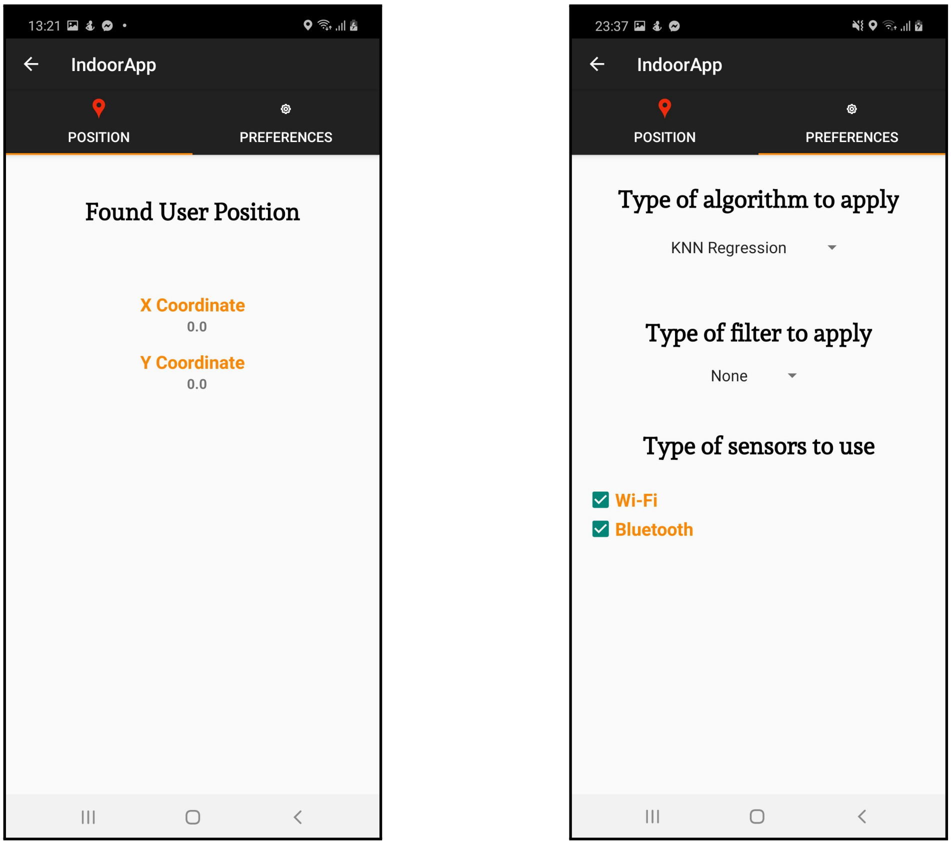
- **Proximity Scanning Screen** (Figure 4.19(a)): This screen allows users to listen for BLE advertisement packets sent from a beacon. Once started, the application listens continuously during a 1 min cycle.

- **Proximity Preferences Screen** (Figure 4.19(b)): This screen enables to configure several parameters of the *offline* scanning procedure.

At the end of each scan interval, the application sends the gathered data to the *IPS Server*. The body of the request contain the retrieved samples with the following information:

- Single RSSI value that holding the last RSSI update captured.

- List of all the RSSI values captured during the cycle to calculate a rolling average.



Found User Position

X Coordinate
0.0

Y Coordinate
0.0

(a) Prediction Screen

Type of algorithm to apply

KNN Regression

Type of filter to apply

None

Type of sensors to use

☑ Wi-Fi
☑ Bluetooth

(b) Preferences Screen

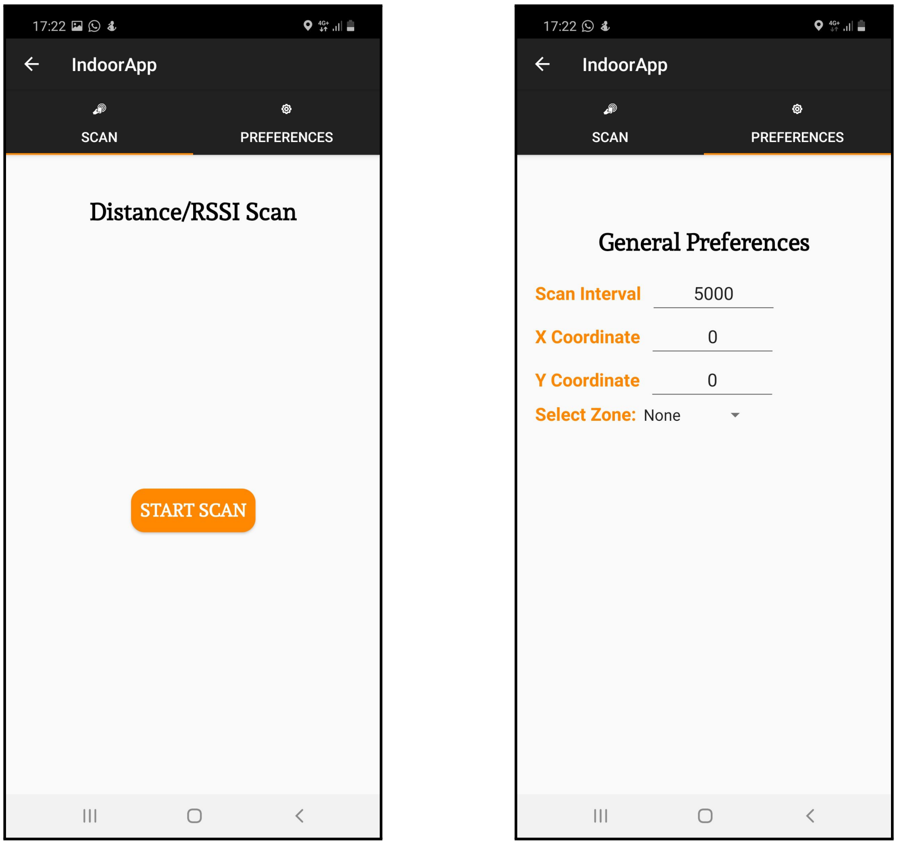
Figure 4.18: *IndoorApp* – *Fingerprinting* Online Screens

- Cartesian coordinates representing the position of the device in relation to the beacon. The *y-axis* coordinate depicts the distance between both of them and the *x-axis* should be ignored.
- The proxemic zone the device is positioned in relation to the beacon.

The *IndoorApp* allows users to visualize the predictions made by the *IPS Server* during the *online phase* in real-time (see Figure 4.20(a)). As depicted in Figure 4.20(b), the user must only configure the machine learning algorithm to be used in the matching phase between the samples saved in the offline data acquisition phase and the data scanned in real-time. The scanning rate differs from the *offline phase* because 1 min would be too long for a user to wait for a position prediction. Instead, the application scans for advertisements of a BLE beacon for 10 s before sending the results to the *IPS Server*.

4.7.3.3 Trilateration

Trilateration takes advantage of the *proximity* technique to predict distances between devices and the beacons placed in the environment. This positioning technique does



General Preferences

Scan Interval 5000

X Coordinate 0

Y Coordinate 0

Select Zone: None

(a) Main Screen

(b) Preferences Screen

Figure 4.19: *IndoorApp* – *Proximity* Offline Screens

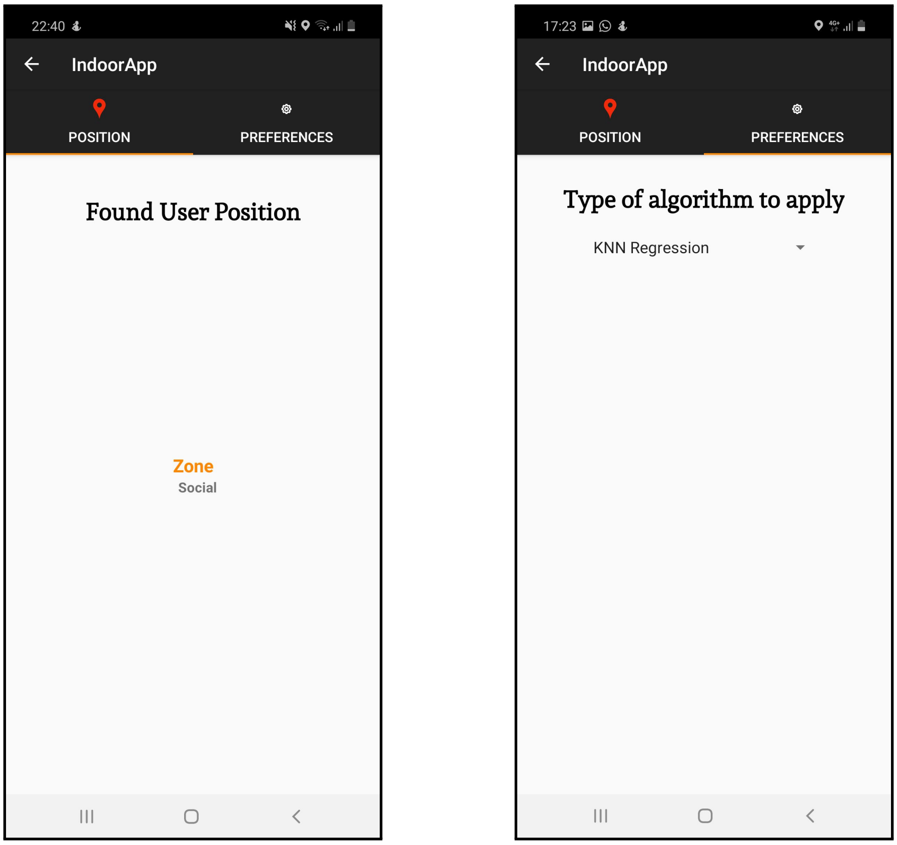
not have an *offline phase* because it relies on the data collected for *proximity* technique. Therefore, the *IndoorApp* only provides a tool to test the *online phase* of this technique. The application offers the following screens (Figure 4.21):

- **Online Prediction Screen** (Figure 4.21(a)): It shows the device position predicted by the *IPS Server* after a scanning cycle of 10 s.

- **Online Preferences Screen** (Figure 4.21(b)): It allows the selection of the machine learning algorithm employed by the underlying *proximity* technique.

4.8 Performance Evaluation

Each indoor positioning technique has multiple variables that can be adjusted. They also have their own advantages and disadvantages when it comes to accuracy and precision. Therefore, we designed a series of experiments to determine the default configuration for each technique and also to determine which technique performs better in certain situations.



Found User Position

Zone
Social

Type of algorithm to apply

KNN Regression

(a) Prediction Screen

(b) Preferences Screen

Figure 4.20: *IndoorApp* – *Proximity* Online Screens

4.8.1 Fingerprinting

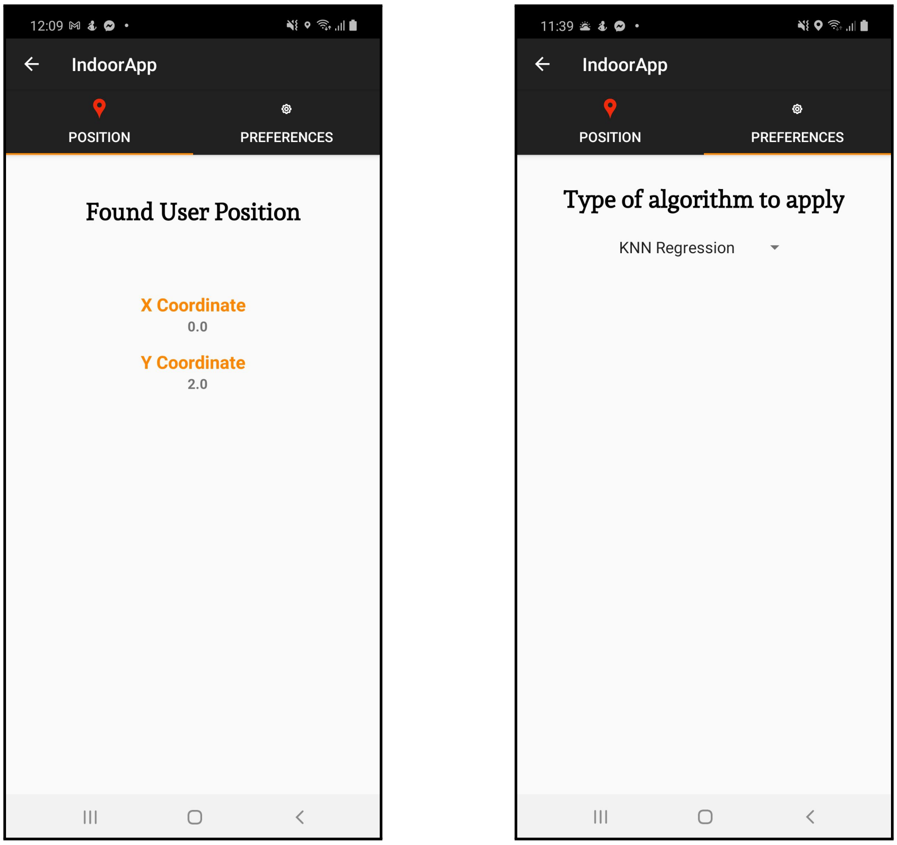
The datasets used for the *fingerprinting* experiments were:

- *Home*: A 2 × 2 m room with 13 reference points and 10 fingerprints scanned per reference point (130 samples in total).
- *University*: 4 × 4 m room with 25 reference points and 30 fingerprints scanned per reference point (750 samples).

4.8.1.1 Evaluation of Machine Learning Algorithms

The online matching phase in *fingerprinting* assesses the similarity between samples scanned in real-time by a device and the fingerprints previously saved in a radio map. Since it is assumed that the fingerprints already have labels associated, supervised learning algorithms are commonly used as part of the matching stage.

Nonetheless, there is no exact answer for which machine learning algorithm provides the best results to predict the position of the devices. Therefore, a set of experiments



Found User Position

X Coordinate
0.0

Y Coordinate
2.0

Type of algorithm to apply

KNN Regression

(a) Prediction Screen

(b) Preferences Screen

Figure 4.21: *IndoorApp* – *Trilateration* Online Screens

were developed to test the following machine learning algorithms: k-Nearest Neighbors (k-NN), Random Forest (RF), SVM, and Multilayer Perceptron (MLP).

The experiment encompassed tuning the hyperparameters of each algorithm using *Stratified K-Fold* cross-validation. The *Leave-One-Group-Out (LOGO)* strategy was also used to simulate the worst-case scenario, i.e., the algorithm is trained with samples from all reference points except the one currently being tested. Table 4.2 shows the comparison between the performance results of the multiple regression machine learning algorithms used to predict the absolute positions of devices while using the *University* dataset. Random Forest is not always the best algorithm but it achieved good performance across all of the experiments. Therefore, it was selected as the default algorithm.

4.8.1.2 Replacement of Missing Values

Fingerprints collected often have missing data for certain access points and beacons because they may appear in a certain place and during a certain scan, and yet be missing from other scans. Hence, the resulting radio map is characterized by fingerprints holding some missing data. As Gerón puts it, “most Machine Learning algorithms cannot work with missing features” [Gér]. Therefore, the missing data must be replaced somehow. This

Table 4.2: *Fingerprinting* – Regression Results

Task	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
Regression	k-NN	0.292113	0.602620	0.909212	1.489585
Regression	SVM	0.548867	0.705709	0.875494	1.496627
Regression	RF	**0.260042**	**0.485201**	**0.941145**	**1.187140**
Regression	MLP	0.879875	1.028073	0.735767	1.875221
Regression (with LOGO)	k-NN	1.462987	1.603255	0.357393	2.597401
Regression (with LOGO)	SVM	1.390924	1.547562	0.401263	2.631749
Regression (with LOGO)	RF	1.394118	**1.508732**	**0.430932**	**2.520780**
Regression (with LOGO)	MLP	**1.390476**	1.515245	0.401356	3.060226

experiment tested replacing the missing values with the *maximum*, *minimum*, *median* or *mean* value of the dataset.

One possible approach is for each missing value to be replaced by a value which is computed globally based on the whole dataset (i.e., *Global Replacement*). Another alternative is for the filler values to be calculated separately for each access point or beacon (i.e., *Replacement by Column*). Hence, it was developed an experiment that evaluated the performance of the two approaches when using the k-NN algorithm. Table 4.3 displays the top 5 performance results of the outlined experiment.

Table 4.3: *Fingerprinting* – Top 5 Strategies to Replace Missing Values

Replacement Strategy	**MAE**	**RMSE**	r^2
Median (Replacement by Column)	**0.167649**	**0.468131**	**0.945213**
Mean (Replacement by Column)	0.168652	0.474131	0.943800
Minimum (Replacement by Column)	0.230479	0.542685	0.926373
Median (Global Replacement)	0.231117	0.540518	0.926960
Mean (Global Replacement)	0.249042	0.576241	0.916987

The best results are obtained with the *mean* or *median* strategies. However, the *minimum* replacement strategy also provides good results and it has a more sound reasoning behind it. Since lower RSSI values should correspond to access points or beacons that are further away, it seems logical that those that are not found during a scanning cycle are represented as being as further away as possible. Therefore, this was the strategy adopted for the remaining experiments and the final solution.

4.8.1.3 Influence of the Environment

Since the system is expected to perform in multiple environments, its performance should be analyzed in at least two different environments. Therefore, we tested the regression performance for multiple machine learning algorithms using the *University* and *Home* datasets. Table 4.4 displays the results of the experiment which show us that radio maps can only be used for predictions in the same environment in which they were trained.

Table 4.4: *Fingerprinting* – Influence of the Environment Results

Environment	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
University	k-NN	0.292113	0.602620	0.909212	1.489585
University	SVM	0.548867	0.705709	0.875494	1.496627
University	RF	**0.260042**	**0.485201**	**0.941145**	**1.187140**
University	MLP	0.879875	1.028073	0.735767	1.875221
Home	SVM	1.445909	1.644998	-0,987927	2.733377
Home	k-NN	1.906570	2.200061	-2.540453	4.197007
Home	RF	1.924238	2.210096	-2,577391	3.924106
Home	MLP	**1.406280**	**1.579692**	**-0,918899**	**2.632281**

4.8.2 Proximity

The *proximity* experiments aim to determine the influence of multiple factors in the quality of the distance predictions. The following datasets were prepared for regression tasks, i.e., to predict the distance between a device and a beacon, and classification tasks, i.e., to determine the proxemic zone relative to a beacon that a device is in:

- *Home Big Dataset*: Radio map with 8 reference points placed between 0 and 3.5 m away from the beacon (12476 samples in total).
- *Home Small Dataset*: Radio map with 8 reference points placed between 0 and 3.5 m away from the beacon (2284 samples in total).
- *Home Different Device Dataset*: Radio map with 8 reference points placed between 0 and 3.5 m away from the beacon (415 samples in total). This dataset was captured with a different device.
- *University Train Dataset*: Radio map with 10 reference points placed between 0 and 4.5 m away from the beacon (24834 samples in total).
- *University Different Beacon Dataset*: Radio map with 10 reference points placed between 0 and 4.5 m away from the beacon (3678 samples in total). This dataset was built using a different *Estimote* beacon.

4.8.2.1 Evaluation of Machine Learning Algorithms

Similarly to *fingerprinting*, the *proximity* technique also applies a machine learning algorithm to match the samples collected during the *online phase* with the data gathered during the *offline phase*. The following algorithms were compared to select the one which performed better: k-Nearest Neighbors (k-NN), *Random Forest* (for classification), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Linear Regression (LR) (for regression).

The experiment followed the same procedure as the corresponding *fingerprinting* experiment. Table 4.5 and 4.6 depict the performance results of the various algorithms in regression and classification tasks using the *University Train Dataset*. As can be seen, k-Nearest Neighbors is not always the best algorithm but it achieved good results across all of the experiments. Therefore, it was selected as the default algorithm.

Table 4.5: *Proximity* – Regression Results

Task	Algorithm	MAE	RMSE	r^2	P_{95}
Regression	k-NN	**0.483617**	**0.723037**	**0.746707**	**1.150000**
Regression	SVM	0.471081	0.806262	0.685040	1.353657
Regression	LR	0.784444	0.960112	0.553372	1.547085
Regression	MLP	0.579432	0.793403	0.694958	1.253388
Regression (with LOGO)	k-NN	**0.945925**	**1.295717**	**0.372113**	1.816667
Regression (with LOGO)	SVM	1.144035	1.402022	0.047464	1.726999
Regression (with LOGO)	LR	1.078450	1.325616	0.148456	**1.722772**
Regression (with LOGO)	MLP	1.190065	1.456349	0.027785	1.985990

Table 4.6: *Proximity* – Classification Results[1]

Alg.	Accuracy	Macro Precision	Macro Recall	Macro F_1 Score	Weighted Precision	Weighted Recall	Weighted F_1 Score
k-NN	0.776137	0.761007	**0.756944**	0.758613	0.778868	0.776137	0.777164
MLP	0.682325	0.752960	0.587338	0.614814	0.740784	0.682325	0.660968
SVM	**0.786203**	**0.784706**	0.746121	**0.761378**	**0.790062**	**0.786203**	**0.784255**
RF	0.780753	0.782356	0.742157	0.757757	0.787430	0.780753	0.779502

4.8.2.2 Dataset Size

The amount of data can significantly impact the performance of a machine learning algorithm. Therefore, this experiment analyzes the impact of the size of the dataset in the accuracy of the predictions. Table 4.7 describes the performance of the distance predictions when using each machine learning algorithm for regression with the *Home Big Dataset* and *Home Small Dataset*. According to the experiment, it seems that collecting more data has a positive impact on how accurately the technique predicts distances.

4.8.2.3 Beacon Dependability

One of the main shortcomings of *fingerprinting* is having to build a radio map for each target environment. With *proximity*, the collected samples are also associated with a singular reference beacon. Therefore, this experiment intends to assess if the samples collected for a given beacon can be used interchangeably to estimate the distance to different beacons, assuming that they are the same model and that they configured with the same parameters, i.e., same signal strength and advertisement rate.

[1] In *Macro*-averaged metrics all classes equally contribute to the final result. In *Weighted*-averaged metrics each classes's contribution to the final result is weighted by its size.

Table 4.7: *Proximity* – Dataset Size Results

Environment	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
Home Big	k-NN	**0.456959**	0.646798	0.680732	1.233333
Home Big	MLP	0.578331	0.711727	0.613415	1.431180
Home Big	SVM	0.731374	0.947848	0.314361	1.750145
Home Big	LR	0.549926	**0.628517**	**0.698524**	**1.071555**
Home Small	k-NN	**0.609939**	0.924156	0.346545	2.250000
Home Small	MLP	0.714681	1.005103	0.227058	2.468916
Home Small	SVM	0.893288	1.171690	0.050390	2.387025
Home Small	LR	0.654051	**0.870916**	**0.419666**	**1.885683**

Table 4.8 presents the performance results of the distance predictions for two *Estimote* beacons (Purple and Blue) scanning samples, depicted in the datasets *University Train Dataset* and *University Different Beacon Dataset*, respectively. There is a loss of performance when using different target beacons with the same radio map. However, the trade-off between the additional work of having to collect data for each specific beacon and the performance loss is generally worth it. Therefore, the system makes predictions based on the same radio map, regardless of the target beacon.

Table 4.8: *Proximity* – Beacon Dependability Results

Beacon	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
Purple	k-NN	**0.483617**	**0.723037**	**0.746707**	**1.150000**
Purple	SVM	0.471081	0.806262	0.685040	1.353657
Purple	LR	0.784444	0.960112	0.553372	1.547085
Purple	MLP	0.579432	0.793403	0.694958	1.253388
Blue	k-NN	0.792349	1.092299	0.415207	1.336667
Blue	SVM	1.082622	1.378281	0.068903	2.250145
Blue	LR	0.804222	**0.986332**	**0.523168**	**1.226325**
Blue	MLP	**0.790095**	1.107381	0.398946	1.775414

4.8.2.4 Influence of the Environment

A possible issue for the *proximity* technique is the possibility that a training dataset can only be used effectively to make online predictions in the same environment where the *offline phase* took place. If that is the case, it becomes much harder for this technique to be successfully used in different environments with minimal effort. Therefore, this experiment addressed this issue by evaluating how different algorithms performed in the *University Training Dataset* and *Home Big Dataset* environments. Table 4.9 outlines the experiments results for the target environments. The results show that a *proximity* training dataset can be applied in different environments. Therefore, a singular radio map can be used for predictions in different indoor environments.

Table 4.9: *Proximity* – Results of Different Environments

Environment	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
University	k-NN	0.483617	**0.723037**	**0.746707**	**1.150000**
University	SVM	**0.471081**	0.806262	0.685040	1.353657
University	LR	0.784444	0.960112	0.553372	1.547085
University	MLP	0.579432	0.793403	0.694958	1.253388
Home	k-NN	**0.456959**	0.646798	0.680732	1.233333
Home	SVM	0.578331	0.711727	0.613415	1.431180
Home	LR	0.731374	0.947848	0.314361	1.750145
Home	MLP	0.549926	**0.628517**	**0.698524**	**1.071555**

4.8.2.5 Device Dependability

RSSI values captured by two devices in the same conditions are susceptible to being different due to variability between devices. This means that a machine learning model may only generalize well to the device used build the training set. If it is used to make predictions for different device it may significantly impact accuracy. This experiment tries to examine how strong is this effect by testing the trained algorithm with a dataset that was captured with the same device used for training (*Home Big Dataset*) and another dataset that was captured with a different device (*Home Different Device Dataset*). The results are presented in Table 4.10). As can be seen, there is a loss of performance with the change of the target device. However, this loss does not pose a significant loss. Therefore, the technique should be perform relatively well across different devices.

Table 4.10: *Proximity* - Device Dependability Results

Device	**Algorithm**	**MAE**	**RMSE**	r^2	P_{95}
Default Device	k-NN	**0.456959**	0.646798	0.680732	1.233333
Default Device	MLP	0.578331	0.711727	0.613415	1.431180
Default Device	SVM	0.731374	0.947848	0.314361	1.750145
Default Device	LR	0.549926	**0.628517**	**0.698524**	**1.071555**
Different Device	k-NN	**0.661165**	0.887742	0.391453	1.750000
Different Device	MLP	0.735414	0.980193	0.258103	2.100547
Different Device	SVM	0.960265	1.200156	0.112233	2.250145
Different Device	LR	0.728780	**0.881859**	**0.399493**	**1.657002**

4.8.3 Trilateration

Trilateration leverages the *proximity* technique to predict the distance between a device and reference beacons in the environment. This means that there is a dependency between the performance of *trilateration* and *proximity*. For *trilateration* itself, the objective was to study the best approach to predict the position of a device based on the distance estimates provided by the *proximity* technique.

The analytical approach of determining the intersection of multiple circles from their equations cannot be used because the inaccuracy of the distance estimates often lead to the definition of circles that do not intercept in a single point. This means that the system of equations usually has no solution. Therefore, the *trilateration* experiments focused on solving the problem from an optimization point of view.

The optimization perspective upon *trilateration* finds the point X that provides the best approximation to the device position P. Specifically, X is the point whose distance to the known beacons best fits the distance between a target device's position (point P) and the beacons in a particular environment. A common approach to achieve this execution flow is to find the point X that minimizes the sum of squared errors, i.e., a least-squares optimization problem.

Consequently, the experiments studied how the following optimization approaches perform:

- **Brute Force Approach**: Finds the point X out of a list of possible P that best fits the estimated distances to each beacon.

- **Optimization Algorithm**: Given the positions of the beacons, use an optimization algorithm that iteratively finds the point X that best fits the estimated distances to each beacon.

Each experiment assumes that the device is at one of the 25 points with saved BLE RSSI samples and will analyze the prediction errors at each point according to the Mean Absolute Error (MAE) metric.

4.8.3.1 Brute Force Approach

The brute force approach finds the point X that minimizes the distance error between X and the set of known beacons, with X being part of a fixed set of available points. This approach goes as follows:

1. Divide the environment into a grid of points.

2. Compute the *Euclidean* distance between each point and the position of the beacons in the environment.

3. Apply the *proximity* technique to predict the distance between the target device and the position of the beacons in the environment.

4. For each point in the grid:

    a) Compute the mean squared error between the point's distances and the device's distances to the known beacons.

    b) Save the point and mean squared error into a dictionary structure.

5. The position of the device will be the point in the dictionary with the minimum mean squared error.

This experiment used the following datasets:

- *University Dataset divided into a grid of* 25 *points*: 4 × 4 m room divided into a grid of 25 points spaced 1 m apart.
- *University Dataset divided into a grid of* 100 *points*: 4 × 4 m room divided into a grid of 100 points spaced 0.5 m apart.
- *University Dataset divided into a grid of* 2500 *points*: 4 × 4 m room divided into a grid of 2500 points spaced 0.1 m apart.

All datasets include the *Euclidean* distance between each point and the position of the beacons.

The experiment analyzed the *trilateration* performance according to the granularity of the datasets. Hence, Table 4.11 displays the brute force experiment results. The *University Train Dataset proximity*'s dataset was used to predict the distance between the target device and the beacons present in the test environment.

Table 4.11: *Trilateration* – Brute Force Results

Dataset	**MAE**
1 m Grid	1.047477
0.5 m Grid	1.044542
0.1 m Grid	**0.948854**

4.8.3.2 Optimization Algorithm

The brute force approach requires the dataset to include information about the candidate points that form the grid pattern. Optimization algorithms can be used as an alternative that only requires the positions of the beacons in the environment. The brute force method is also pretty naive and gets increasingly more computationally expensive as the number of points on the grid increases. Therefore, we experimented with the following libraries that implement least-squares optimization:

- **SciPy Minimize Function**[The21]: The *scipy.optimize.minimize* function was used to obtain a result by applying a bound-constrained optimization algorithm (*L-BFGS-B* [Byr+95; Zhu+97]) with the sum of the squared error as the target error function to be minimized [Vir+20]. The initial prediction of the function is the target device's nearest beacon.
- **Localization package**[Kam21]: This *Python* package applies *trilateration* by trying to find a point that minimizes the sum of squared distance errors to the position of

the devices. To achieve this, the system only requires the distance of the target device to known beacons and their respective positions. The underlying optimization algorithms are also provided by *SciPy* [Vir+20].

Table 4.12 outlines the performance results, with the best brute force result as a baseline. The two implementations that were tested performed identically. Therefore, the *SciPy Minimize Function* was selected since it is part of well established library which offers more options to explore to further improve the results.

Table 4.12: *Trilateration* - Results

Procedure	**MAE**	**RMSE**	P_{95}
Localization Package	1.041392	1.140057	1.895565
SciPy Minimize	1.041344	1.140028	1.895580
Brute-force	**0.948854**	**1.131371**	**1.882843**

4.8.4 Comparison of Positioning Techniques

Since *proximity* only provides relative positions, it cannot be directly compared with *fingerprinting* and *trilateration*, which provide the absolute location of a device. Absolute positions offer more information and can also be used to infer the relative positions in respect to other devices and PoIs. Moreover, although *fingerprinting* present good overall results, its performance results in the worst-case scenario (*LOGO*) degrade considerably. Therefore, *trilateration* presents itself as the positioning technique with the good performance and more consistent results. It also has the added advantage of requiring considerably less data collection during the *offline phase*.

5

Application Prototypes

As part of the development of the *YanuX Framework*, we have continuously tested each of its components, both isolated and integrated according to their dependencies. We have also started to create prototypes to test and evaluate our solution in controlled environments.

5.1 YanuX YouTube Viewer

The *YanuX YouTube Viewer* is our first application prototype. It plays *YouTube* videos and spreads its UI elements across multiple devices by synchronizing the application state between them. Its source code can be found at: `https://github.com/YanuX-Framework/YanuX-YouTubeViewer`. It uses the *YouTube IFrame Player API* to load and control the playback of a video given its *URL* [Goo18c]. The selected video takes the center stage on the application if only one device is available, as can be seen in Figure 5.1.

Each UI component is highlighted with a dark dashed border. From top to bottom, the *viewer-form* component has a text field where users should input a *YouTube* video *URL* and a submit button to load the video. Next is the *player* component, which is where the video is played back. Finally, the *controls* component provides a seek bar to jump to any point of the video, and displays the total and current playback times. The buttons below the seek bar are for seeking the video backwards 5 s, playing it, pausing it, stopping it or seeking it 5 s forward, respectively.

The application was built taking advantage of the *YanuX Coordinator* and using *YanuX Auth*'s *OAuth 2.0 Authorization Server* for user authentication and client application authorization. The *YanuX Coordinator* connects to the *YanuX Broker*, and retrieves the user specific global application and proxemic states. It will keep getting notified of any changes made to them. It will also watch for any other custom events defined by the application developer.

For the *YanuX YouTube Viewer*, the global application state stores the *ID* of the *YouTube* video currently loaded, its playback state (e.g. playing, paused or stopped), current time and duration. This state is continuously updated once per second as the video is playing,

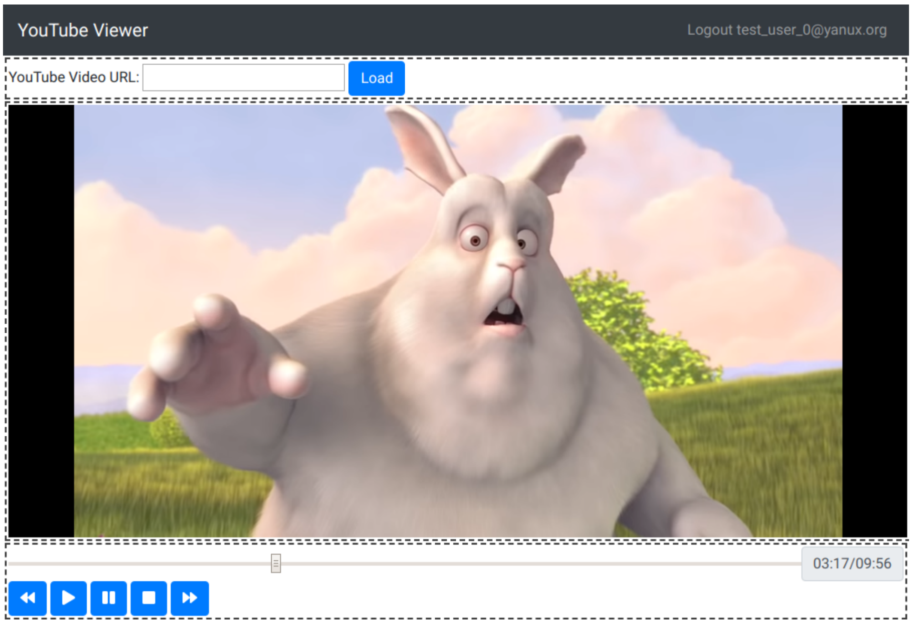


Figure 5.1: *YanuX YouTube Viewer* displaying all the components in a single device configuration.

but it can also be changed if the user loads a new video, or if one of the playback control buttons is pressed.

We follow a simple pattern to deal with user interaction under most circumstances. We locally modify the global state when a user performs some interaction that should have an impact on the state of the application's UI elements. Those changes are immediately submitted to the *YanuX Broker* through the *YanuX Coordinator*. Afterwards, all connected clients (including the one that submitted the changes) will get the new global state, which should be used to update their UI.

However, there are situations that do not fit well into that pattern. For instance, a custom *seekTo* event is used instead of the usual pattern to deal with the manipulation of the seek bar and the two seek buttons. We identified that a concurrency issue would arise whenever there were more than a single device playing back the video.

A *player* component which got synced to the new selected time, would correctly propagate the change on its 1 s update schedule. However, components which had yet to react to the change would continue to propagate the outdated playback time on their 1 s update cycle. This would lead to the *player* components incorrectly going back and forth the two playback times.

A custom event solves the problem because it is only triggered once by the device used to interact with the application. It will then be received and interpreted once by each of the connected devices, which will use the *YouTube IFrame Player API* to seek to the time in the event's payload.

Regarding the automatic distribution of the UI, the requirements for each of the components of the applications had to be defined. The first step was ensuring that the

YanuX Orchestrator submitted the device's characteristics to the *YanuX Broker*.

The other important step is the requirements definition for the UI components of each application by using our *JSON*-based *DSL*. In the case of the *YanuX YouTube Viewer*, Listing 5.1 presents the definition of the requirements for the three UI components of the YanuX YouTube Viewer prototype: *viewer-form*, *controls* and *player*.

Listing 5.1: UI components' requirements for the application

```
{
  "viewer-form": {
    "type": { "value": "smartphone", "enforce": false },
    "display": true,
    "input": {
      "operator": "OR", "values": [
        { "operator": "AND", "values": ["keyboard", "mouse"] },
        "touchscreen"
      ]
    }
  },
  "controls": /*The same as "viewer-form".*/,
  "player": {
    "display": {
      "operator": "AND", "values": {
        "resolution": { "operator": ">=", "value": [960, null] },
        "size": {
          "operator": ">=", "value": [160,90], "enforce":false
        }
      }
    },
    "speakers": {
      "channels": {
        "operator": "AND", "values": [
          { "operator": ">=", "value": 2, "enforce": false },
          { "operator": ">=", "value": 1 }
        ]
      }
    }
  }
}
```

We have focused on a scenario in which we intend for larger screens, such as a smart TV, only displaying the *player* component (Figure 5.2). Meanwhile, the *viewer-form* and *control* should be left to smaller and more personal devices like a smartphone (Figure 5.3).

We have restricted the *player* component to be placed on a screen with a resolution of at least 960 pixels in width, and a physical size of at least 160 by 90 mm. We are indicating

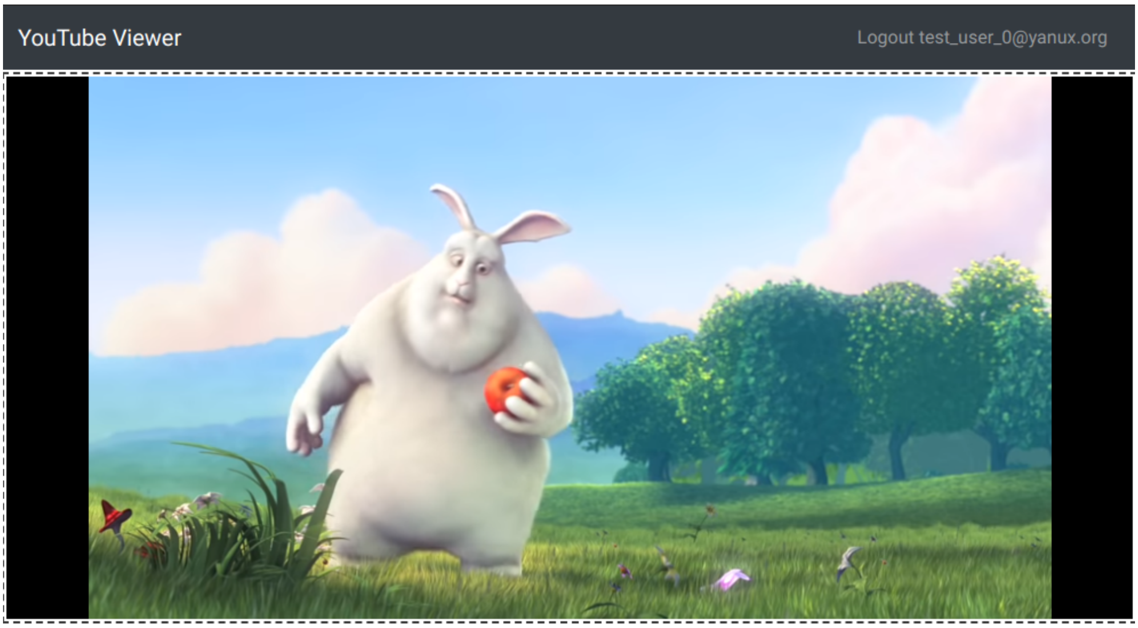


Figure 5.2: *YanuX YouTube Viewer* only displaying the *player*.

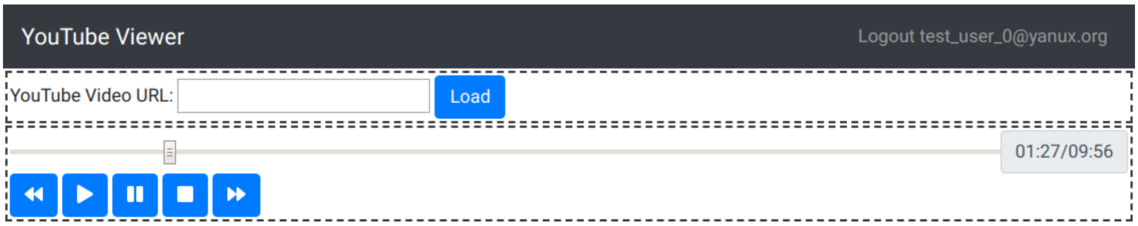


Figure 5.3: *YanuX YouTube Viewer* displaying the *viewer-form* and *controls* components.

that both requirements must be met by grouping them together in an *operator: AND* block. However, the size requirement is relaxed thanks to the *enforce: false* declaration. It means that this restriction will be disregarded if there is no other device present that is able to display the *player* component.

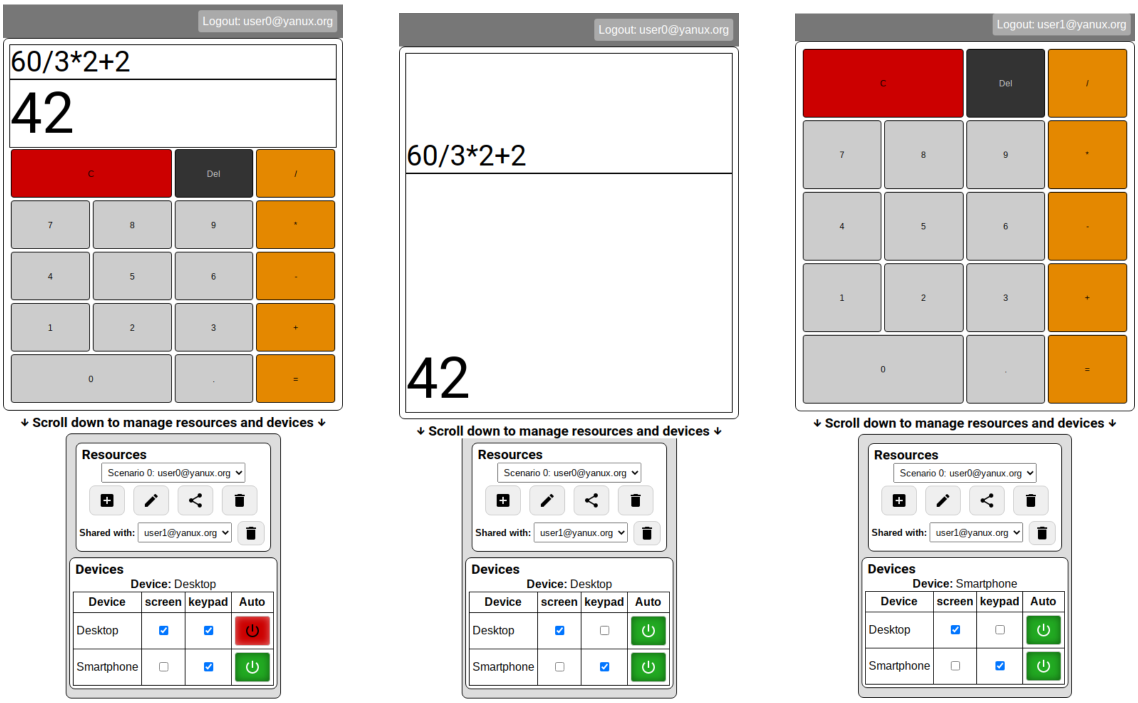
There are also other requirements which are placed on the devices' *speakers*, besides the ones already placed on the *display*. We declare that we want a device with at least 2 audio channels for stereo sound, but we do not enforce this requirement as we did for the device size. The one that must absolutely be met is the requirement for one or more audio channels, which ensures that the *player* component is only placed on devices that can playback the audio portion of the videos.

The *viewer-form* and *control* have the same requirements. We prefer to place them on a smartphone device, but do not enforce it, which must have a display without having any specific restrictions. However, the device needs a way to receive user input in the form of keyboard and mouse. Alternatively, the two components can also work with touchscreen input.

This configuration allows a cross-device experience when accessing the *YanuX YouTube Viewer* through a web browser on devices running the *YanuX Orchestrator* and the *IPS Client* in the background. As the devices detect each other's presence, the UI will adapt itself according to the restrictions we have defined. All actions will be propagated between the devices and all UI components will remain synchronized to provide a cohesive user experience.

5.2 YanuX Calculator

The *YanuX Calculator* is a cross-device collaborative calculator and its source code is available at: `https://github.com/YanuX-Framework/YanuX-Calculator`. The UI of the calculator is split into two components (see Figure 5.4): a *screen* with the currently inserted arithmetic expression and the result; and a *keypad*. They can be shown in the same device (Figure 5.4(a)) or separately (Figures 5.4(b) and 5.4(c)). Either way, they completely fill the screen. If users scroll down they find the *YanuX Resource Management* and *YanuX Components Distribution* elements.



Shared with: user1@yanux.org

Device: Desktop

Device	screen	keypad	Auto
Desktop	☑	☑	⏻
Smartphone	□	☑	⏻

(a) All components.

Shared with: user1@yanux.org

Device: Desktop

Device	screen	keypad	Auto
Desktop	☑	□	⏻
Smartphone	□	☑	⏻

(b) Only the *screen* component.

Shared with: user1@yanux.org

Device: Smartphone

Device	screen	keypad	Auto
Desktop	☑	□	⏻
Smartphone	□	☑	⏻

(c) Only the *keypad* component.

Figure 5.4: Possible UI distributions of the *YanuX Calculator* application.

The *YanuX Resource Management Element* allows users to manage application states, i.e., *resources*. They are used to save multiple expressions and results. Those *resources* can be shared to enable collaboration. The *YanuX Components Distribution Element* displays the distribution of components across application instances. It allows users to change the distribution or to let it be determined based on the proxemic relationships of the devices running those instances.

The application was built using *React* and *Redux*. User authentication and the authorization is provided by *YanuX Auth*. The *YanuX Coordinator* is used to connect to the *YanuX Broker* to retrieve *resources* representing the application state, proxemic relationships between devices and information about active application instances. It is used to manipulate entities stored by the *YanuX Broker* while it keeps listening for relevant changes made by other clients.

For instance, since any user interaction causes changes in the *Redux* store, we registered a middleware that uses the *YanuX Coordinator* to submit changes made to the store to the *YanuX Broker* in order to be saved to the currently selected *resource*. Connected clients can then get events with the information needed to keep their own *Redux* stores in sync.

Regarding the automatic distribution of the UI components, we envisioned a scenario where the *screen* component is displayed on larger screens which primarily serve as output devices (e.g., large monitor or TV). The *keypad* component should be placed on personal devices which perform the role of input devices (e.g., a smartphone or tablet).

These two situations correspond to Figures 5.4(b) and 5.4(c), respectively. We then used our *JSON*-based *DSL* to define the requirements for each of the UI components according to this scenario (see Listing 5.2).

Listing 5.2: The requirements of *YanuX Calculator*'s UI components

```json
{
  "screen": {
    "showByDefault": true,
    "display": {
      "operator": "AND",
      "values": {
        "virtualResolution": {
          "operator": ">=", "value": [1024, null]
        },
        "size": {
          "operator":">=", "value": [160, 90], "enforce": false
        }
      }
    }
  },
  "keypad": {
    "showByDefault": true,
    "type": { "value": "smartphone", "enforce": false },
    "display": true,
    "input": { "operator": "OR", "values": ["mouse", "touchscreen"] }
    }
}
```

The property *showByDefault* was set to *true* for both the *screen* and *keypad*. This indicates that the components should be displayed if there is no other active application instance running on a device which matches the component's requirements. This ensures there will be at least one copy of each component even if they have to be placed suboptimally.

The *screen* component should be placed on a device with a *display* with, at least, a *virtual resolution*[1] of 1024 px in width and at least a *size* of 160 mm in width by 90 mm in height. Both requirements are grouped with an *AND* operator, but the *size* requirement is marked as not enforced. Therefore, the algorithm tries to match both requirements, but it will settle for just the *virtual resolution* if there is no active instance running on another device that matches both requirements.

The *keypad* component should run on devices classified as *smartphone*, but this requirement is marked as not enforced. The algorithm will give preference to smartphones but it will drop the requirement if there is no smartphone available. We also defined that the *keypad* component requires a *display* of any kind, and a *mouse* or a *touchscreen* for *input*.

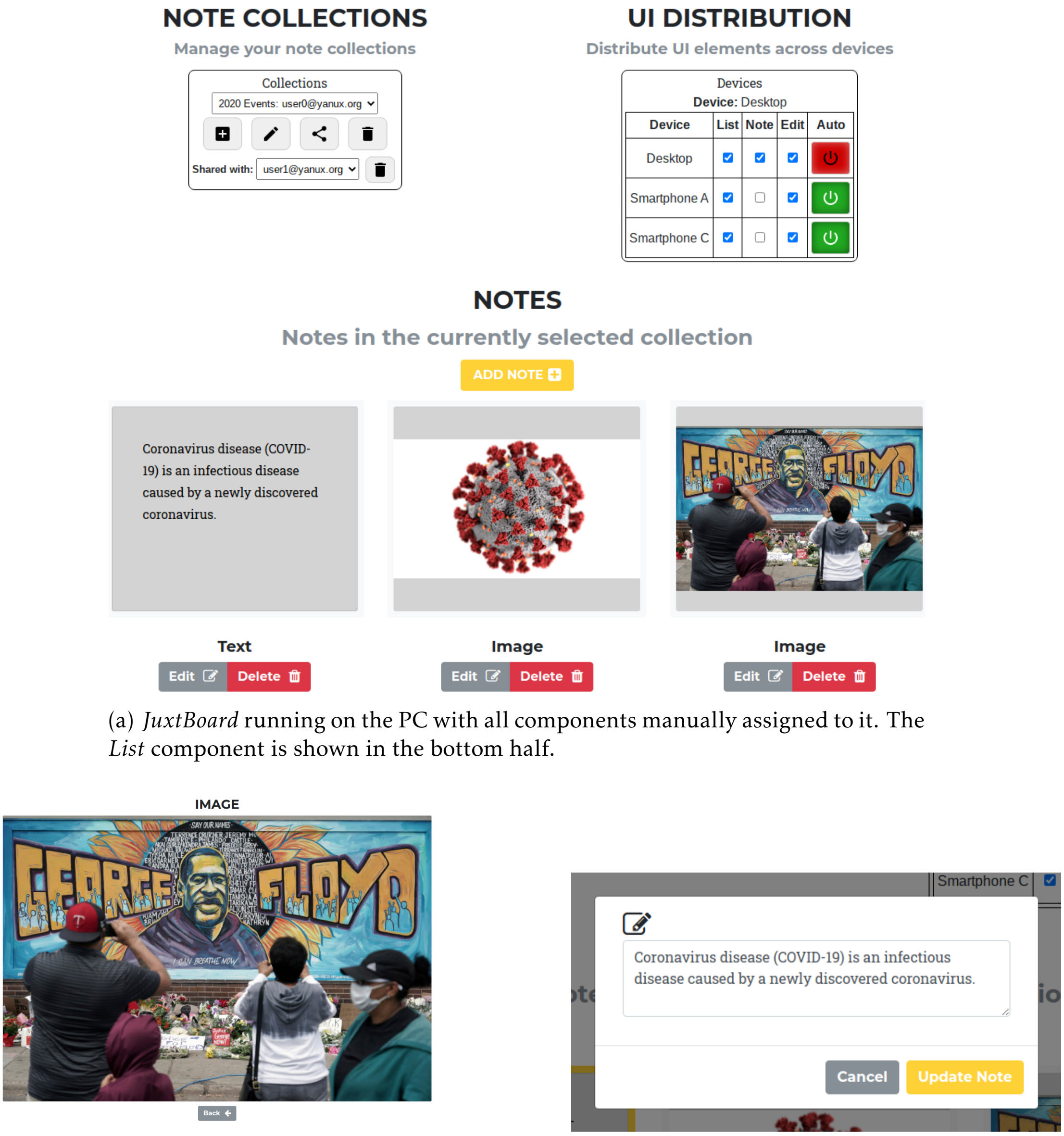
5.3 JuxtBoard

The *JuxtBoard* application was also built using *React*. It is a digital board where users can add text, image or video notes. Those notes are placed into collections created by users. The collections can also be shared in other to allow other users to collaboratively add, edit or remove notes. The source code of the application is available at: `https://github.com/YanuX-Framework/YanuX-JuxtBoard`

The UI of the application has three main components which can be distributed by multiple devices: the *List* component which lists the notes and shows a preview of each one (bottom half of Figure 5.5(a)), the *Note* component which displays the contents of a note in full-screen (Figure 5.5(b)), and the *Edit* component displays a form to edit the text of note, or to replace the image or video associated with a note with another one (Figure 5.5(c)). The components are also responsive in order to adapt to screens of different sizes. For instance, if the *List* component needs to be displayed on smartphone, it will show all of the note previews on a single column.

The application uses the YanuX Coordinator to get and manipulate information on the YanuX Broker and to receive events triggered by other clients, such as changes made to the *resources* representing the collections of note, or information about the available application instances and which components should be displayed on each one.

[1] Virtual resolution is based on pixel density just like *CSS pixels* in Web browsers



NOTE COLLECTIONS

Manage your note collections

Collections

2020 Events: user0@yanux.org

Shared with: user1@yanux.org

UI DISTRIBUTION

Distribute UI elements across devices

Devices

Device: Desktop

Device	List	Note	Edit	Auto
Desktop	☑	☑	☑	⏻
Smartphone A	☑	☐	☑	⏻
Smartphone C	☑	☐	☑	⏻

NOTES

Notes in the currently selected collection

ADD NOTE

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.

(a) *JuxtBoard* running on the PC with all components manually assigned to it. The *List* component is shown in the bottom half.

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.

(b) The *Note* component.

(c) The *Edit* component.

Figure 5.5: The *JuxtBoard* applications and its components.

It also receives events indirectly triggered by the YanuX IPS Bridge. As it submits the positioning information to the YanuX Broker, new proxemic relationships are established or broken. This generates events to which application instance can also subscribe to through the YanuX Coordinator.

The *YanuX Resource Management Element* allows users to manage their *resources* (see the left side of the top half of Figure 5.5(a)). Each *resource* stores metadata about a collection and the contents of the notes are stored by a separate application service. The *YanuX Resource Management Element* can be used to share *resources* with multiple users so that they can work collaboratively on the same collections.

The *YanuX Components Distribution Element* displays the distribution of components across application instances (see the right side of the top half of Figure 5.5(a)). It allows users to change the distribution or to let it be determined based on the proxemic relationships of the devices running those instances.

We used *React*'s *Context API* to manage the local application state by following a similar pattern to the one used by *Redux*. Therefore, we made sure that any relevant changes made locally are intercepted and sent through the YanuX Coordinator to the YanuX Broker to be stored in the *resource* representing the currently selected collection. Other connected clients will then get events to keep their local state synchronized.

The scenario envisioned for the automatic distribution of UI components was to let users manage notes using the *List* component and to use the *Edit* component to edit notes on personal devices (e.g., a smartphone or tablet). If a large display is available, the *Note* component would be displayed there for a better experience and to foster collaboration. We used our *JSON*-based *DSL* to define the requirements for each of the UI components according to the desired scenarios as can be see Listing 5.3.

Listing 5.3: The requirements of *JuxtBoard*'s UI components

```
{
  "List": {
    "type": { "value": "smartphone", "enforce": false},
    "display": true,
    "input": { "operator": "OR", "values": ["mouse", "touchscreen"] }
  },
  "Note": {
    "display": {
      "operator": "AND", "values": {
        "virtualResolution": {
          "operator": ">=", "value":[1024, null], "enforce": false
        },
        "size": {
          "operator": ">=", "value": [160,90], "enforce": false
        }
      }
    },
    "speakers":true
  },
  "Edit": {
    "type": { "value":"smartphone", "enforce": false },
    "display":true,
    "input": {
      "operator": "AND", "values": [
        { "operator": "OR", "values": ["mouse", "touchscreen"] },
        { "operator": "OR",
          "values": ["touchscreen", "keyboard", "speech"] }
      ]
    }
  }
}
```

The *List* and *Edit* components have similar requirements. They are primarily meant to be shown on *smartphone* devices, but this is not enforced. This means that that any device will display the component if there is no other device that meets the ideal requirements. Both require a *display* and *mouse* or *touchscreen* for input. However, the *Edit* component additionally requires a *keyboard*, *touchscreen*, or *speech* input because the user also needs to insert text.

The *Note* component requires a *display* with at least a *virtual resolution* of 1024 px in width and at least a size of 160 mm in width by 90 mm in height. These requirements are marked as not enforced, which means that any device will display the component if there is no other device that meets the requirements. However, when larger devices that meet these requirements are available the component will be preferably be shown on them. The component also requires *speakers* in order to play audio from video notes.

The *YanuX Orchestrator* is used to collect device capabilities. Our experimental setup consisted of a PC running *Linux* to a large display and two *Android* smartphones. Table 3.1 shows a summary of the capabilities collected for the three devices that were used by *JuxtBoard*.

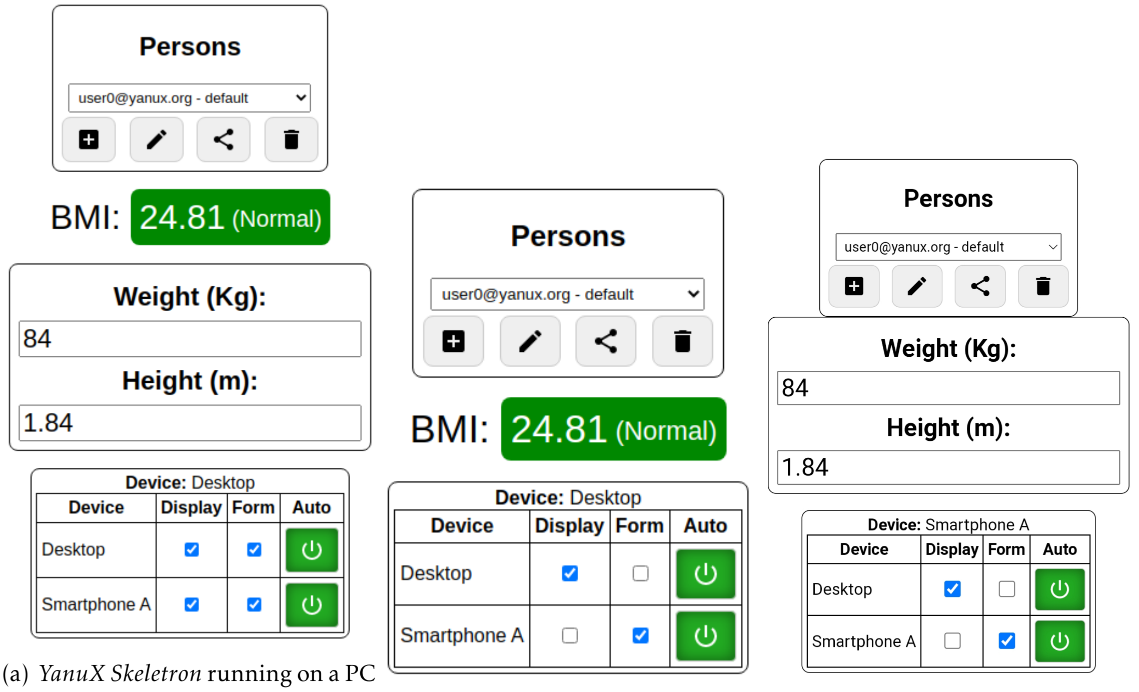
The prototype was made available as a Progressive Web App which (PWA) which can be accessed through a Web browser as long as the *YanuX Orchestrator* and the *IPS Client* are running in the background of each of the devices. As devices detect each other, the UI will adapt according to the restrictions that were defined and user interactions will be propagated across devices thanks to the *YanuX Coordinator* and the rest of the framework and of the indoor positioning system.

5.4 YanuX Skeletron

The *YanuX Skeletron* application was built to be used as part of a user study with developers. Therefore, it was purposely built without the use of any additional frameworks or libraries, i.e., it only uses plain *JavaScript* on a single file and our own YanuX Coordinator library. The application itself was also kept simple to avoid exposing the user study participants to a complex code base on top of the concepts introduced as part of the *YanuX Framework*. Its source code can be found at: `https://github.com/YanuX-Framework/YanuX-Skeletron`.

The application implements a Body Mass Index (BMI) calculator which consists of two UI components: a *Display* that shows the BMI value and a *Form* where a user can enter the weight in kilograms and the height in meters (see Figures 5.6(b) and 5.6(c), for a scenario where each component is shown on different devices). The *Display* states if the

BMI value corresponds to an underweight person (< 18.5), normal (≥ 18.5 and < 25) or overweight (≥ 25). Its color also changes to *blue*, *green* and *red*, respectively. Figure 5.6(a) shows a scenario in which both components are being displayed in the same device.



Persons

user0@yanux.org - default

BMI: 24.81 (Normal)

Weight (Kg): 84

Height (m): 1.84

Device: Desktop

Device	Display	Form	Auto
Desktop	☑	☑	⏻
Smartphone A	☑	☑	⏻

(a) *YanuX Skeletron* running on a PC with all components visible. The *Display* is shown on top of the *Form* component.

Persons

user0@yanux.org - default

BMI: 24.81 (Normal)

Device: Desktop

Device	Display	Form	Auto
Desktop	☑	☐	⏻
Smartphone A	☐	☑	⏻

(b) *YanuX Skeletron* running on a PC with the *Display* component visible.

Persons

user0@yanux.org - default

Weight (Kg): 84

Height (m): 1.84

Device: Smartphone A

Device	Display	Form	Auto
Desktop	☑	☐	⏻
Smartphone A	☐	☑	⏻

(c) *YanuX Skeletron* running on a smartphone with the *Form* component visible.

Figure 5.6: The *YanuX Skeletron* applications and its components.

Thanks to the *YanuX Framework* the state of both components is always kept in sync across the application instances running on multiple devices that are accessing the a *resource* representing the same application state. Moreover, the *YanuX Resource Management Element* and the *YanuX Components Distribution Element* are always shown on top and on the bottom of the displayed components (see Figure 5.6). In the case of the *YanuX Resource Management Element*, it is used to manage the multiple application states in the form of *resources*. In this application each *resource* stores the weight, height and BMI of a person. Therefore, this *web component* was personalized to reflect that using the *slots* that it makes available to the developers [WHA21]. As usual, the *YanuX Components Distribution Element* is used to show the current distribution of UI components, allowing users to change the distribution manually and to set it back to automatic on a per device basis.

For the YanuX Skeletron the requirements for the automatic distribution of UI components were kept relatively simple on purpose because we intended to show participants of the developer-focused study how they could use the DSL to implement the scenario they were testing. We envisioned a scenario where the *Display* component is shown on larger screens which primarily serve as output devices, e.g., a monitor or TV (see Figure 5.6(b)). Meanwhile, the *Form* component should be placed on a personal devices, e.g., a smartphone or tablet, which perform the role of an input device for the other component (see Figure 5.6(c)). However, when only a device is available, or the two devices are not associated with each other according to their proxemic relationships (e.g., they are not close enough or properly oriented), the two components will be shown in the available devices (see Figure 5.6(a)).

The requirements needed to implement the aforementioned scenario are defined in Listing 5.4 using the DSL that is part of the *YanuX Framework*.

Listing 5.4: The requirements of *YanuX Skeletron*'s UI components

```
{"Display":{
   "display":{"virtualResolution":{"operator":">=",
                                   "value":[1024, null],
                                   "enforce":false}}
 },
 "Form": {"display":true,
          "input":{"operator":"OR",
                   "values":["mouse","touchscreen"]},
          "type":{"value": "smartphone", "enforce": false }}
}
```

The *Display* component has a restriction on the *display* of the device where it is shown. It should have at least a *virtualResolution*[1] of 1024 px in width. However, since this requirement is marked as *enforce: false*, its enforcement is relaxed, i.e., a device that does not meet the criteria will display the component if there is no other device that meets the requirement. This is done to ensure that users always have a way to check the value of the *BMI*, even if it has to be shown on a suboptimal device.

The *Form* component requires that a device has a *display*, that it has *mouse* or *touchscreen* available for *input*, and also that it should be placed on a *smartphone*. However, this last requirement is marked as not enforced (i.e., *enforce: false*). It means that it will be displayed on other types of devices if no smartphones are available.

[1] Virtual resolution is based on pixel density just like *CSS pixels* in Web browsers

6

User Studies

The previously presented prototypes were used to conduct a series of user studies. Table 6.1 summarizes the capabilities collected for the devices used in the user studies. It is important to note that unused information has been omitted for brevity. The *YanuX YouTube Viewer* only used the first two devices, i.e., the *PC* and *Smartphone 1*. The other two prototypes used all of the devices.

Table 6.1: Summary of the device capabilities

Capability	PC	Smartphone 1	Smartphone 2
Type	PC	Smartphone	Smartphone
Display			
Resolution	1920x1080	1080x2248	1080x1920
Pixel Density	96[1]	402	403
Size	5080x286[1]	68x142[1]	68x121[1]
Virtual Resolution	1920x1080[1]	403x839[1]	402x715[1]
Input	Keyboard, Mouse, Speech Recognition	Touchscreen, Speech Recognition	Touchscreen, Speech Recognition
Speakers	Yes	Yes	yes

The prototypes can be accessed through a Web browser as long as the *YanuX Orchestrator* and the *IPS Client* are running in the background. As devices detect each other, the UI will adapt according to the restrictions that were defined and user interactions will be propagated across devices.

6.1 YanuX YouTube Viewer

We conducted a user study in order to evaluate the *YanuX YouTube Viewer* and the receptivity to our vision.

6.1.1 Participants and Design

We had a total of 15 participants (6 female) aged from 22 to 59 years old ($\mu = 28.47, \sigma = 9.20$). The tests took place at our University campus, so it is not unexpected that 14

[1] Property inferred from related properties because it could not be directly retrieved.

participants had a Bachelor's degree or higher education. We acknowledge this sample is not representative of the general population. However, we still believe that our results may generalize well for this population group and that may be indicative of a more general trend.

Participants were briefed about the goals of our research. We asked them to interact with the application and to move multiple times between two distinct usage scenarios. First, they were introduced to a single device usage scenario, in which they only used the application on an *Android* smartphone that had the *IPS Client* and *YanuX Orchestrator* running in the background. Participants could load videos and interact with the controls. This first scenario intended to mimic the way most applications work nowadays.

Afterwards, we asked them to move to a cross-device scenario by moving towards a desk, place the smartphone on top of it, and sit in front of a display connected to a laptop running the application, with the *IPS Client* and *YanuX Orchestrator* in the background. This forced the application to distribute the UI components across the two devices by placing the *viewer-form* and *controls* components on the smartphone, and the *player* components on the display. The users were asked to interact with the application, notice that the state was kept synchronized between the devices, and verify if all operations could still be performed.

They were asked to do the opposite and move way from the display. This time they should have noticed the *player* component moving back to the smartphone and disappearing from the display. We then asked the participants to move back to the front of the display and to notice what happened when they turned off the smartphone's screen. This time, despite the smartphone's presence, the *viewer-form* and *controls* components would appear on the display since the smartphone's application instance was inactive.

Finally, we let users continue to explore the system as they wished and asked them to fill out a questionnaire with demographic questions, the System Usability Scale (SUS) [Bro96], the NPS [Rei03] question, and their level of agreement with the following domain specific statements (7-point *Likert scale*):

- *Q1* – I often use two or more of those devices at the same time.
- *Q2* – Distance is a good measure of how closely related two devices are.
- *Q3* – I found that the time it takes for the application to react to my input to be adequate.
- *Q4* – I found the time it takes for the application to detect that I am close to the monitor to be adequate.
- *Q5* – I found the time it takes for the application to detect that I moved away from the monitor to be adequate.
- *Q6* – I found the distribution of UI components among devices to be adequate.

The complete questionnaire is presented in Appendix A.

6.1.2 Results

We obtained a SUS score of 91.89. This value is above the median of 68, which is considered to be the minimum for an application to have an acceptable level of usability [LS18]. In fact, it falls within the range of *Excellent* usability rating and corresponds to an *A* grade [BKM09].

We also calculated the NPS score to be 66.7%. The distribution of the responses can be seen in Table 6.2 and in Figure 6.1. This metric is more business oriented, which does not map directly to our situation. However, it indicates that the vast majority of our users felt confident that the system was interesting enough to recommend to their friends and family. Nonetheless, we should not be too optimistic about the SUS and NPS scores given the limited scope of our tests and the demographic bias already discussed.

Table 6.2: *YanuX YouTube Viewer* User Study: Statistics summary of the NPS question.

1	0.0%
2	0.0%
3	0.0%
4	0.0%
5	0.0%
6	9.1%
7	13.3%
8	20.0%
9	26.7%
10	40.0%
NPS	66.7
Promoters	66.7%
Passives	33.3%
Detractors	0.0%
Median ($\bar{x}$)	9.0
Mean ($\tilde{x}$)	8.93
SD (s)	1.10

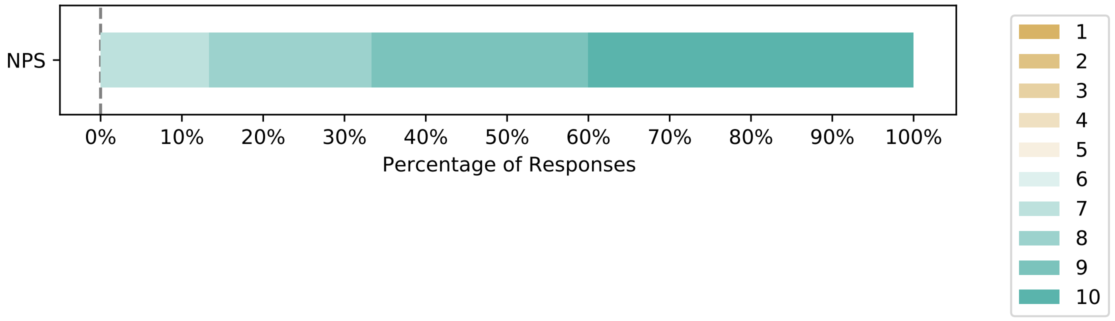


Figure 6.1: *YanuX YouTube Viewer* User Study: Response distribution of the NPS question.

Table 6.3 and Figure 6.2 summarize the answers to our six domain specific questions, presenting the relative frequency of the level of agreement (*1 - Strongly Disagree* to *7 - Strongly Agree*), the mean of the answers ($\bar{x}$) and the standard deviation (s). The results are generally good with most users having a high level of agreement with our statements, corroborated by the mean of each statement's answers being close to 6 (minimum of 5.87) with relatively small standard deviations (maximum of 1.19).

Table 6.3: *YanuX YouTube Viewer* User Study: Statistics summary of the application specific questions

	1	**2**	**3**	**4**	**5**	**6**	**7**	$\bar{x}$	s
Q1	0.0%	6.7%	0.0%	0.0%	20.0%	33.3%	40.0%	5.93	1.33
Q2	0.0%	0.0%	0.0%	6.7%	20.0%	53.3%	20.0%	5.87	0.83
Q3	0.0%	0.0%	6.7%	0.0%	0.0%	33.3%	60.0%	6.40	1.06
Q4	0.0%	0.0%	6.7%	0.0%	13.3%	33.3%	46.7%	6.13	1.13
Q5	0.0%	0.0%	6.7%	6.7%	13.3%	40.0%	33.3%	5.87	1.19
Q6	0.0%	0.0%	0.0%	6.7%	20.0%	46.7%	26.7%	5.93	0.88

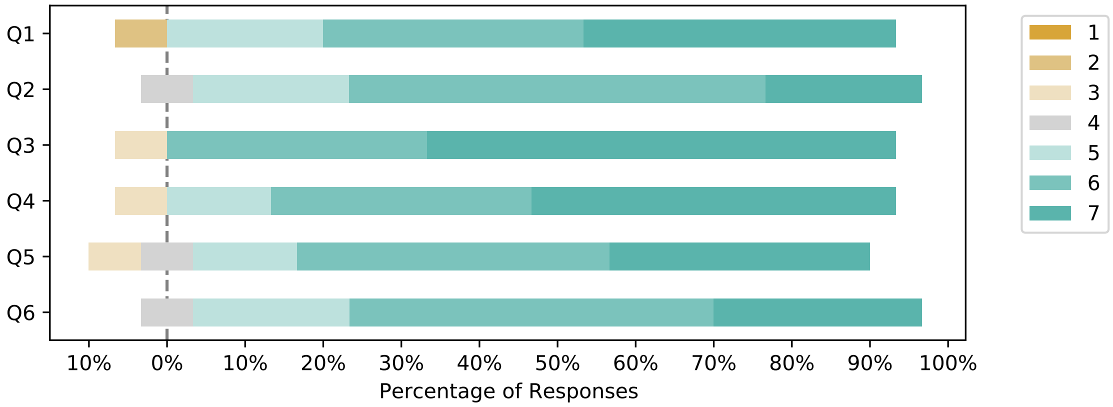


Figure 6.2: *YanuX YouTube Viewer* User Study: Response distribution of the application specific questions.

From the results of *Q1*, we can clearly see that most people already use more than one device at the same time, even thought they are still limited in what they can do with them. Moreover, as part of our questionnaire we also asked what devices the users owned. All of them owned a smartphone and a laptop, and on average they own at least 4 devices.

The answers to *Q2* tell us that using distance between devices as a measure of relatedness was a good idea, although we may want to explore other forms of association in the future. *Q3* and *Q4* shows that our solution for indoor positioning can meet user expectations. Finally, the results of *Q6* confirm that users generally enjoyed the distribution that was chosen based on the information about co-located devices and the restrictions that developers placed on the components.

6.2 YanuX Calculator

We conducted an user study using the *YanuX Calculator* application to continue to evaluate how users receive the concepts promoted by our framework.

At this time, the *YanuX Components Distribution Element* was one of the latest additions to the framework. Therefore, we were particularly interested in understanding whether users could easily visualize the components distribution and change the automatic distribution according to their needs.

The other addition was the *YanuX Resource Management Element*, so we wanted to assess if users could grasp the concept of multiple application states (*resources*), how to manage them and how to share them with others to enable collaboration.

6.2.1 Participants and Design

We had a total of 13 participants (5 female) aged from 20 to 60 years old ($\mu = 31.85, \sigma = 12.79$). The tests took place at our University campus, so the participants either already had a Bachelor's degree or higher education, or were enrolled in a Bachelor's degree program. Therefore, we acknowledge that this sample is not representative of the general population, but we had difficulty in gathering more participants due to the COVID-19 pandemic. In any case, we consider our results may generalize well for this population group which may be indicative of a more general trend.

The participants were briefed about our research and the application. They were told that they had to collaborate with another user to perform some tasks. The role of the other user was performed by one of the researchers. The test setup consisted of a PC connected to a monitor and two smartphones (see Table 6.1). Each participant was placed in front of the PC and was given a smartphone logged in with the same user account. The researcher would take the other smartphone, which was logged in with a separate account, and sit a couple of meters away, but with the PC monitor still clearly visible.

Before going through task scenarios, we allowed the participants to explore the UI of the application with minimal guidance. At this stage, the PC monitor would only show the *screen* component like in Figure 5.4(b) and the smartphone of the participants would show the *keypad* like in Figure 5.4(c). Afterwards, the participants went through three scenarios in which they had to use the application in collaboration with the researcher to solve basic arithmetic problems.

Each participant was required to create a resource at the start of each scenario and to share it with the user account of the researcher. As long as the newly shared resource is selected by the researcher on his own device, the devices owned by both user accounts will be used to create a shared interaction environment.

After going through the three scenarios, participants were asked to move several meters away from the PC monitor and to take the smartphone with them. The researcher accompanied them and asked them to take notice of how the UI was redistributed, with the smartphones now showing both components like in Figure 5.4(a) because there is no longer a large screen in the vicinity to display the *screen* component. Therefore, the smartphones fell back to showing the *screen* component on a sub-optimal display.

Participants were then told to move back to the initial position, in order to take notice that the initial distribution was automatically reestablished. They could then experiment with the manual distribution of UI components and with resetting the distribution back to automatic. They were also instructed to rename, delete and unshare resources.

Finally, we asked them to fill out a questionnaire with demographic questions and their level of agreement with nine application specific statements (see Table 6.4). Afterwards, they answered to the SUS to quickly assess the usability of the prototype [Bro96]. We also wanted them to respond to the UEQ in order to measure the user experience (UX) of our application according to six different scales [LHS08; SHT14]. The complete questionnaire is presented in Appendix B.

6.2.2 Results

We now present a summary of the responses to some domain specific questions. Table 6.4 presents the relative frequency of the level of agreement (*Likert scale* from *1 - Strongly Disagree* to *7 - Strongly Agree*), the median ($\tilde{x}$), the mean ($\bar{x}$) and the standard deviation (s) of the answers. The results are generally good, with a high level of agreement with the statements.

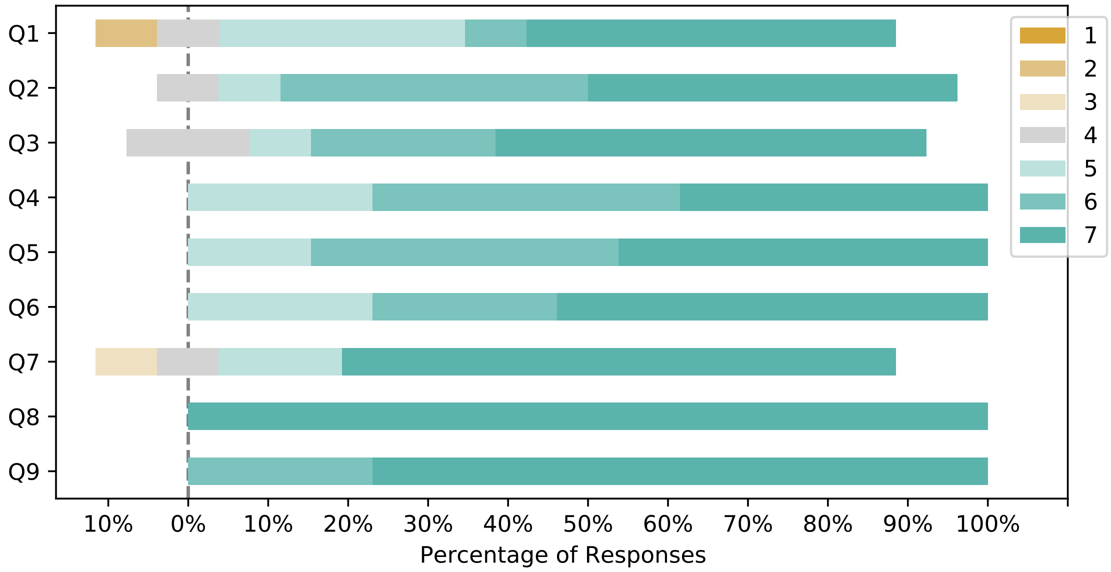


Figure 6.3: *YanuX Calculator* User Study: Response distribution of the domain specific questions.

According to *Q1*, participants already have a tendency to use more than one device at the same time. This is a hint that there may be an opportunity to expand upon what users

Table 6.4: *YanuX Calculator* User Study: Statistics summary of the domain specific questions

1	**2**	**3**	**4**	**5**	**6**	**7**	$\tilde{x}$	$\bar{x}$	s
Q1: I often use two or more devices at the same time in my daily life.									
0.0%	7.7%	0.0%	7.7%	30.8%	7.7%	46.2%	6.0	5.69	1.55
Q2: I found that the time it takes for the application to react to my input to be adequate.									
0.0%	0.0%	0.0%	7.7%	7.7%	38.5%	46.2%	6.0	6.23	0.93
Q3: Distance is a good measure of how closely related two devices are.									
0.0%	0.0%	0.0%	15.4%	7.7%	23.1%	53.8%	7.0	6.15	1.14
Q4: I found the time it takes for the shared display to react to my presence to be adequate.									
0.0%	0.0%	0.0%	0.0%	23.1%	38.5%	38.5%	6.0	6.15	0.80
Q5: I found the automatic distribution of UI components among devices to be adequate.									
0.0%	0.0%	0.0%	0.0%	15.4%	38.5%	46.2%	6.0	6.31	0.75
Q6: It is easy to change the distribution of UI components to suit my needs.									
0.0%	0.0%	0.0%	0.0%	23.1%	23.1%	53.8%	7.0	6.31	0.85
Q7: I could easily figure out how to change back the UI distribution to be automatically distributed.									
0.0%	0.0%	7.7%	7.7%	15.4%	0.0%	69.2%	7.0	6.15	1.41
Q8: It was easy to add, rename or remove resources (saved application states) from the application.									
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100%	7.0	7.0	0.0
Q9: I could easily give and remove access to my resources (saved application states) to other users.									
0.0%	0.0%	0.0%	0.0%	0.0%	23.1%	76.9%	7.0	6.77	0.44

are already doing naturally with the new possibilities opened by the *YanuX Framework*. Moreover, the answers to *Q3* confirm that the distance between devices can be used to measure if they are related.

The results of *Q2* and *Q4* are indicative that our system reacts to input according to user expectations and that our indoor positioning solution can keep up with the changes in the proxemic relationship of co-located devices. The automatic distribution of UI components was also within expectations according to *Q5*. Meanwhile, *Q6* and *Q7* indicate that it was easy to use the *YanuX Components Distribution Element* to change the components distribution and that users discovered how to set the distribution back to automatic. Nevertheless, we received feedback about making the *Auto* buttons more predictable and suggestions to improve visual cues.

Questions *Q8* and *Q9*, which focused on managing and sharing application states, i.e., *resources*, got the best results. This implies that users found the *YanuX Resource Management Element* quite easy to use. Nevertheless, we received feedback on how to improve it. For instance, some users expected that after creating a *resource* the application would automatically switch to it. Participants also brought attention to some minor issues that are worth addressing.

The results from the UEQ were also very encouraging (see Figure 6.4). We got values above 1.5 in every scale, with most of them surpassing 2.0. These are very positive results given the theoretical range of each scale starts at −3, as the most negative score, goes up to 3, as the most positive score, and that 0.8 is the minimum value for considering an evaluation as positive [Sch19].

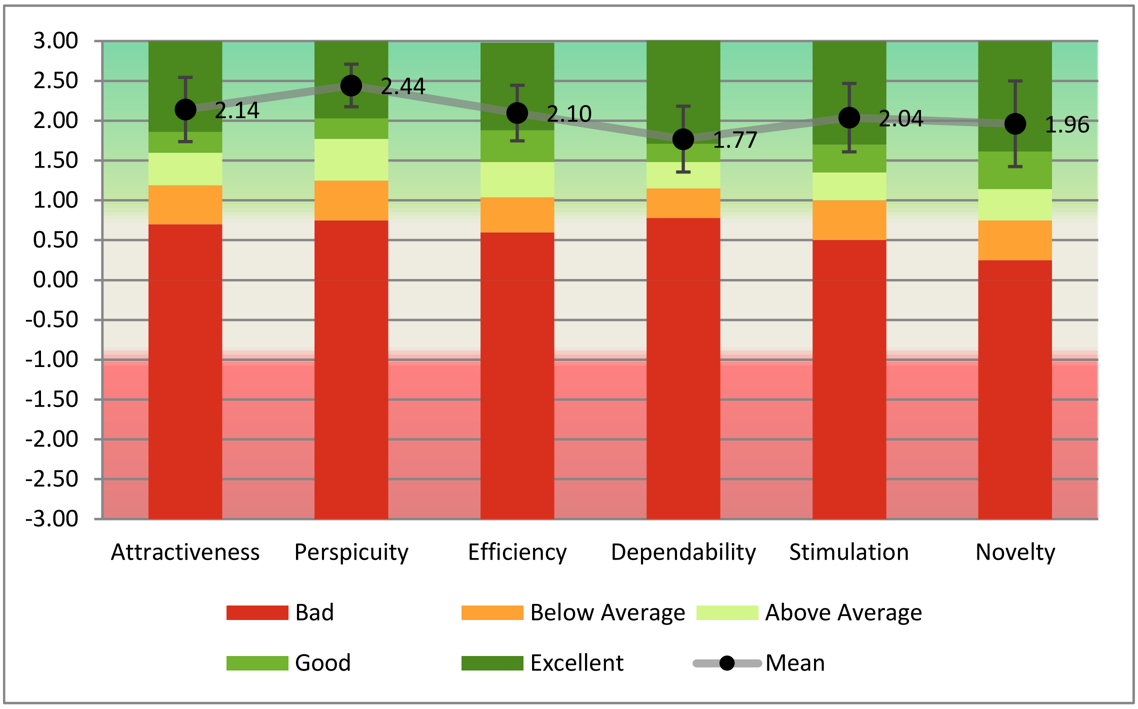


Figure 6.4: *YanuX Calculator* User Study: The results for the six different UEQ scales.

Our prototype scored very high in *Perspicuity*, which focuses on how easy it is to get familiar with the product and to learn how to use it. This score is very important to us, because we introduce new concepts that are not usually found in other applications. Conversely, the scale in which we scored the worst was *Dependability*. Putting the prototype application aside, this may indicate that we need to improve the way the framework allows users to manage multiple devices, application states and how they collaborate with others to provide additional control and predictability.

It is also worth noting that the scores in all six scales are considered *Excellent* and are in the top 10% percent results according to the UEQ benchmark [Sch19]. Even if we consider the error bars representing the 95% confidence intervals the results still correspond to a *Good* grade, except in the *Dependability* scale which may fall to *Above Average*. Nonetheless, we are aware that we should not be too optimistic about the SUS and UEQ results given the limited scope of our tests and the already discussed demographic bias.

We obtained a SUS (System Usability Scale) [Bro96] score of 87.69. This value is well above the threshold of 70, which is generally considered to be the minimum for an acceptable level of usability. It is within the range of an *Excellent* usability rating corresponding to a solid *B* grade [BKM09]. Even if we consider the lower bound of the

95% confidence interval ($[80.43, 94.95]$) it can be considered a *Good* rating and the *B* grade still holds.

6.3 JuxtBoard

We conducted another user study with the *JuxtBoard* application to continue to evaluate how users receive the concepts promoted by our framework when using a more realistic application. Moreover, there were also major improvements made to the Indoor Positioning System (IPS) and many other adjustments that were made to the framework that needed to be tested by users.

6.3.1 Participants and Design

For the *JuxtBoard* user study we had 27 participants (4 female) aged from 19 to 60 (median age of 20, mean of 23.48, and standard deviation of 9.61). The study took place in a University campus, so the majority of the participants were enrolled students. We acknowledge this sample is not representative of the general population, but the results should generalize well and can be an indicative of a general trend. We consider this population as a typical driver for the implantation in society of innovative approaches regarding the interaction between humans and computational systems.

The room where the user study took place is represented in Figure 6.5. The participants were briefed about our research and the application. Each participant (blue user named *P* in Figure 6.5) was told s/he had to collaborate with another user to perform some tasks. A researcher took upon the role of the other user (red user named *R* in Figure 6.5).

Each participant sat on a desk that was in front of a large display, which had a dedicated *BLE* beacon placed on top. The display showed the *JuxtBoard* application that was running on the connected laptop next to it. Each participant was given a smartphone logged in with the same user account as the one being used on the large display. The researcher sat besides the participant with another smartphone which was logged in with a separate account so that collections of notes could be shared between different users.

The tasks were performed twice to compare the application in single device mode, i.e., using only a smartphone to display all components, and in cross-device mode, i.e., using the smartphones to display the *List* and *Edit* components and the large display to show the *Notes* component. To eliminate the effect of any bias towards one of the interaction modes, half of the participants tried the single device mode first and the other half began with the cross-device mode. In both cases, the participants were supposed to collaborate by showing, talking about what they were doing, and sharing collections with the researcher as they performed the tasks.

The participants remained within the *Zone A* marked on the right side of Figure 6.5 while they were performing those tasks in either the single device and cross-device mode.

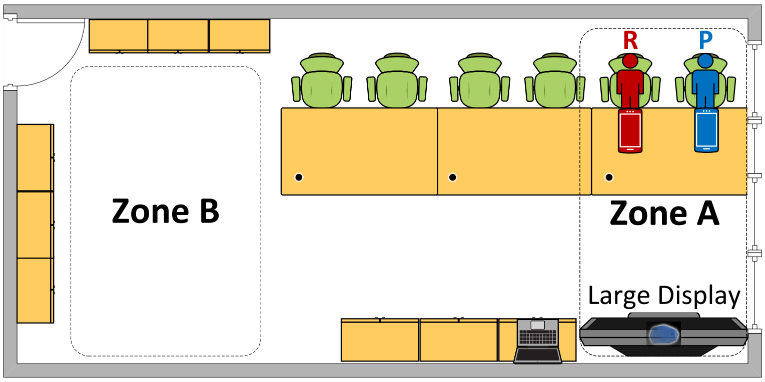


Figure 6.5: Room configuration in the *JuxtBoard* user study.

After they had gone through the tasks in cross-device mode, they were asked to move away from that zone in front of the large display to the *Zone B* on the left size of Figure 6.5. The researcher accompanied them and asked them to take notice of the UI redistribution that took place. Similarly, they were also told to remain in *Zone A* but to face away from the display while holding their smartphone because orientation is also taken into consideration.

In both cases, the *Note* component started to be shown on the smartphones, because they were either too far away from the display or no longer facing it. Participants were then told to move back to their initial positions in *Zone A*, in order to take notice that the initial distribution was automatically reestablished. They were then told to experiment with the manual distribution of UI components and with the possibility of resetting the distribution back to automatic for each one of the devices. At the end, they filled a questionnaire about their experience using the application so that we could gather data and feedback as can be seen in Appendix C.

During the user study, we used the *IPS Server* in *proximity sensing* mode, i.e., the estimates of the distance between devices returned by this positioning technique were used to aggregate the devices instead of inferring those distances from absolute positions. The orientation was also taken into consideration. The *IPS Client* was both used to scan and emit *BLE* beacon signals, except for the PC that relied on a physical beacon near the display, as pictured in Figure 6.5, instead of emitting its own signal.

This decision was taken to simplify the deployment of our test environment, because building a radio map for *fingerprinting*, or placing and calibrating *BLE* beacons for *trilateration*, was not required. Moreover, our scenario only required relative distances. Besides, the estimates returned by *proximity sensing* were more stable and accurate than relative distances calculated from absolute positions provided by *fingerprinting* or *trilateration*.

The issue is probably due to the fact that we are dealing with two absolute position estimates, each one already with some associated error, which gets magnified when the euclidean distance between two points is calculated. Therefore, the proximity based approach was selected because it should lead to a better user experience for this application until some further improvements are made to deal with these issues.

6.3.2 Results

Most participants (84.6%) admitted that sometimes they use more than one device simultaneously. This is a hint that there may be an opportunity to innovate based on what users are already doing with the new possibilities opened by the *YanuX Framework*.

We present a summary of the responses to the domain specific questions that were part of our questionnaire. Table 6.5 and Figure 6.6 present the relative frequency of the level of agreement (*Likert* items from *1 - Strongly Disagree* to *7 - Strongly Agree*), the median ($\tilde{x}$), the mean ($\bar{x}$) and the standard deviation (s) of the answers. The results are generally good, with a high level of agreement with the statements.

Table 6.5: Summary of Domain Specific Questions

1	**2**	**3**	**4**	**5**	**6**	**7**	$\tilde{x}$	$\bar{x}$	s
Q1: The time it takes for the application to react to my input is adequate.									
0.0%	0.0%	0.0%	0.0%	7.4%	40.7%	51.9%	7.0	6.44	0.64
Q2: Distance is a good measure of how closely related two devices are.									
0.0%	0.0%	0.0%	0.0%	7.4%	40.7%	51.9%	7.0	6.44	0.64
Q3: The time it takes for the shared display to react to my presence is adequate.									
0.0%	0.0%	0.0%	3.7%	3.7%	44.4%	48.1%	6.0	6.37	0.74
Q4: The automatic distribution of UI components across devices is adequate.									
0.0%	0.0%	0.0%	7.4%	7.4%	37.0%	48.1%	6.0	6.26	0.90
Q5: It is easy to change the distribution of UI components to suit my needs.									
0.0%	0.0%	3.7%	3.7%	14.8%	37.0%	40.7%	6.0	6.07	1.04
Q6: It was easy to add, rename, remove, share and unshare collections of notes in the application.									
0.0%	0.0%	0.0%	0.0%	0.0%	22.2%	77.7%	7.0	6.78	0.42

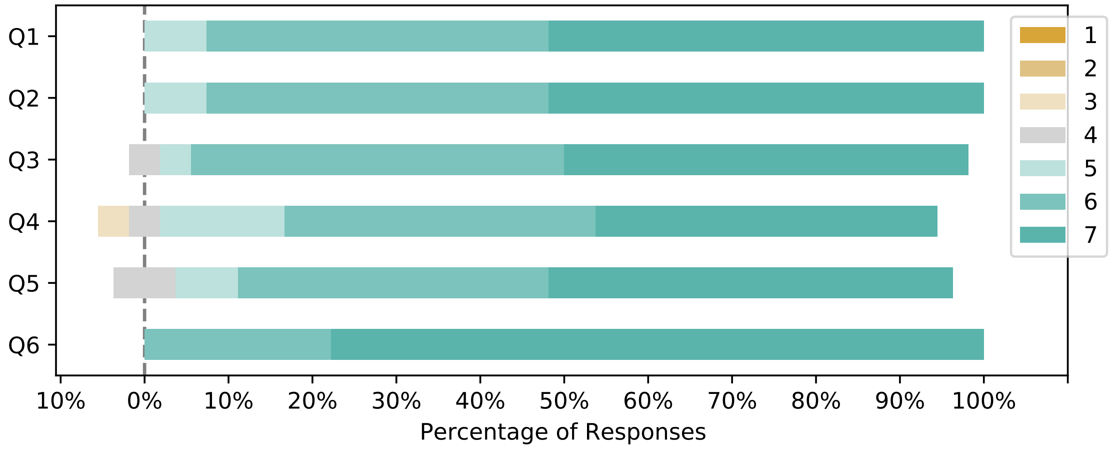


Figure 6.6: *JuxtBoard* User Study: Response distribution of the domain specific questions.

The answers to *Q1* indicate that the framework upon which the application is built is responsive enough for an interactive application. Similarly, the answers to *Q3* indicate that our indoor positioning solution can keep up with the changes in the proxemic relationship of co-located devices. Moreover, according to the answers to *Q2*, the participants consider that the distance between devices is a good way of determining if they are related.

The way the automatic distribution of UI components behaves was also within expectations according to the answers to *Q4*. This hints that users understood and received well the concept introduced by our framework of distributing user interface components across devices based on their characteristics and on their physical relationships.

Regarding the answers to *Q5*, they indicate that the *YanuX Components Distribution Element* (previously seen in Figure 3.4) was easy to use to manipulate the distribution of components. During the user study we noticed that most users were able to discover how to set the distribution back to automatic using the corresponding button for each device. Nevertheless, we received feedback about making the *Auto* buttons more predictable and to improve their visual cues to make them easier to use and understand.

Q6 focused on the management and sharing of collections, which in the context of the *JuxtBoard* application are represented by *resources* stored on the *YanuX Broker* as saved application states. This management was implemented by integrating the *YanuX Resource Management Element* (previously seen in Figure 3.3) into the application. Given the positive results of the answers to this question, it seems that users found this custom reusable element quite easy to use. The feedback we received was also positive, with only suggestions being made to improve some minor details.

We also included the 10 standard questions in our questionnaire that are part of the *System Usability Scale* (*SUS*) for each of the tested interaction modes and calculated the corresponding scores [Bro96]. The *SUS* score was 85.68 for the single-device mode and 89.87 for the cross-device mode.

Both scores are within the range of an *Excellent* usability rating and a *B* grade, with the cross-device mode being close to the *A* grade [BKM09]. The *SUS* scores also hold well even if we consider the 95% confidence intervals of $[83.31, 88.04]$ and $[86.53, 93.22]$, respectively. This leads us to believe that the regardless of the interaction mode the provided experience was well within the expectations that the participants had for an digital pin board application of this type, given the limited time they had to use and explore the application during our user study.

Despite the close results of the two interaction modes, the results show that users have a consistent tendency towards preferring the cross-device mode. We performed a paired sample t-test to assess that tendency and the p-value that we obtained was just 0.003 which is well below the common threshold of 0.05 required to reject the null hypothesis, which in this case is that there is no difference between the *SUS* scores of the single-device and cross-device modes.

Participants were also asked to directly rate their preference between the two interaction modes they experienced according to a series of statements. In this case, increasingly negative values up to −3 meant that the single device mode was absolutely the best and increasingly positive values up to 3 meant the same for the cross-device mode. The neutral value of 0 means that there is no preference between the two modes. Table 6.6 and Figure 6.7 summarize the responses to these questions.

Table 6.6: Summary of Single Device vs. Cross-device Mode

-3	**-2**	**-1**	**0**	**1**	**2**	**3**	$\tilde{x}$	$\bar{x}$	**s**
Q1: In which mode was it easier to manage the collections of notes in the application (e.g., selecting, adding, sharing, renaming or removing collections of notes)?									
0.0%	3.7%	0.0%	59.3%	18.5%	14.8%	3.7%	0.0	0.52	1.01
Q2: In which mode was it easier to manage the notes in the application (e.g., selecting, adding, renaming or removing notes)?									
0.0%	3.7%	0.0%	59.3%	11.1%	18.5%	7.4%	0.0	0.63	1.15
Q3: In which mode was it easier to collaborate and show notes to other users?									
0.0%	0.0%	3.7%	3.7%	3.7%	22.2%	66.7%	3.0	2.44	1.01
Q4: What is your degree of preference between the two modes when using the application to perform tasks alone?									
22.2%	33.3%	3.7%	14.8%	7.4%	0.0%	18.5%	-2.0	-0.74	2.18
Q5: What is your degree of preference between the two modes when using the application to perform tasks in collaboration with other users?									
0.0%	0.0%	3.7%	3.7%	3.7%	14.8%	74.1%	3.0	2.52	1.01

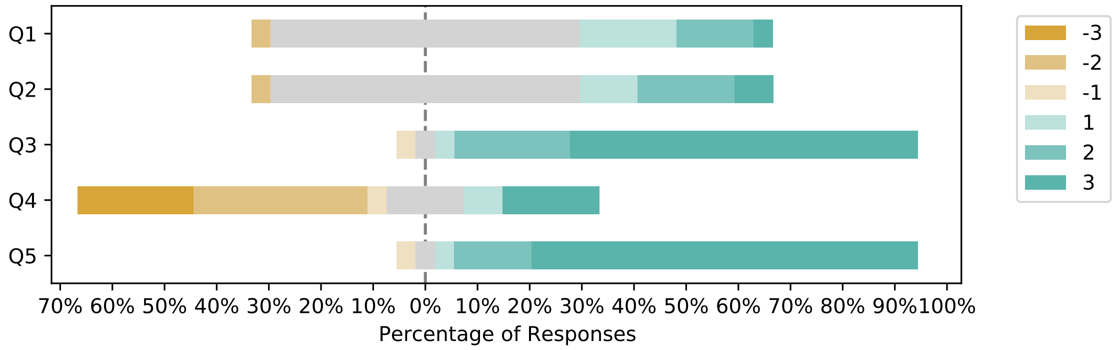


Figure 6.7: *JuxtBoard* User Study: Response distribution of the Single Device vs. Cross-device Mode questions.

As part of our user study, we wanted to assess if the added complexity of the cross-device mode would make it more difficult for people to use the application. However, *Q1* and *Q2* do not give a clear advantage to any of the interaction modes, with most participants considering it to be as easy to perform basic tasks on one mode as on the other.

We believe that adding cross-device interaction capabilities to existing applications should not significantly hinder their usability. At least, as long as it is done carefully so that it is not confusing or working in unexpected ways. This means that the components that can be distributed, and their requirements, should be defined early in the development of cross-device applications to make any necessary adjustments before advancing further with a problematic choice.

Q3 and *Q5* were asked to assess if users found the addition of more devices to be advantageous in situations in which they needed to collaborate and share what they were doing with other users in their surroundings. The responses gave a clear advantage to the cross-device scenario for this type of usage, hinting that this is a situation in which this type of experience is more necessary. Conversely, *Q4* shows us that when users are alone they tend to use a single device.

Nevertheless, there is still a significant percentage of users that chose 0 (neutral) or 3 (absolute preference towards cross-device). This means that some users may also value using multiple devices on their own to have more screen real estate which should provide a better experience. In the particular case of *JuxtBoard*, users are able to comfortably view images and videos on the large display on front of them instead of relying solely on the smaller screen of their smartphone even if they are alone in the room.

6.4 Developer-focused Study with YanuX Skeletron

The final user study that was conducted focused on evaluating the *YanuX Framework* from the perspective of the developers that could potentially use it to build applications with the UI distributed across multiple co-located devices. The YanuX Skeletron application prototype was used as the basis for this user study (see section 5.4).

6.4.1 Participants

There were 22 participants that took part of the user study (4 of them were female). They were aged from 20 to 52 years old ($\mu = 26.36$, $\sigma = 6.93$, $Median = 24$) for the developer-focused study with the YanuX Skeletron. Figure 6.8 provides an overview of the demographics and background of the participants.

The study took place in a University campus but, since we were mostly interested in the feedback from potential users of our framework, we made sure that vast majority of our participants were postgraduate or final-year students of a bachelor degree in Computer Science. There were also some participants with a higher professional technical course in programming and a final-year student from a bachelor degree in Electrical Engineering. This sample represents some of the potential users of our framework, but we acknowledge that it could be expanded to include participants with a stronger professional background.

Participants answered several questions to characterize them. Table 6.7 presents a

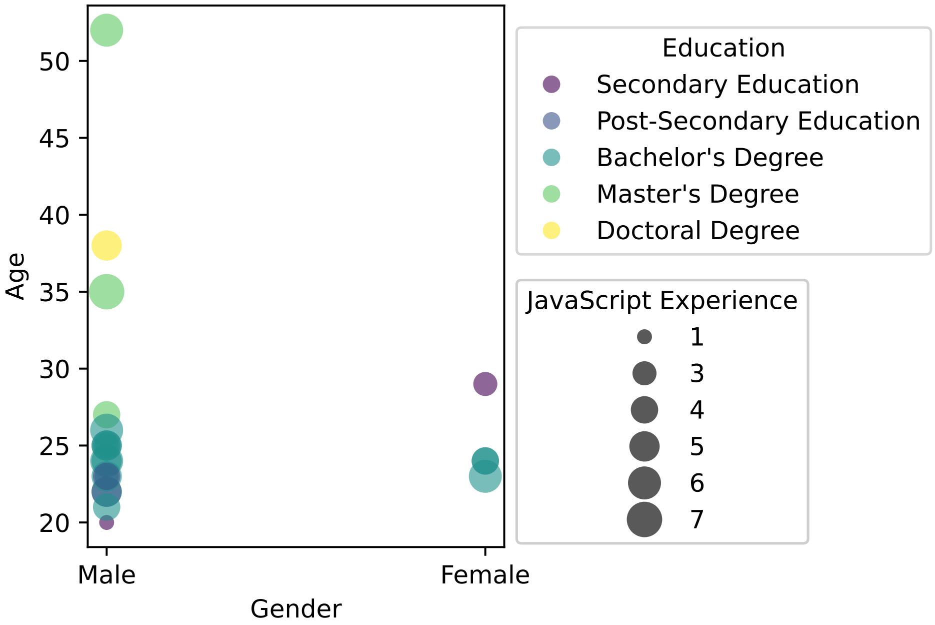


Figure 6.8: Demographics and background of the participants (age, last concluded education level, *JavaScript* programming experience)

summary of several statistics regarding the demographics and background of the participants that took part of the study.

Table 6.7: Statistics of the participants in the user study

	μ	σ	**Median**	**IQR**
Age (Years)	26.36	6.93	24.00	2.75
Years of Programming Experience	8.50	6.67	6.50	3.00
JavaScript Experience (1 - Beginner/7 - Expert)	4.55	1.30	5.00	1.00
Number of Known Programming Languages	6.00	2.59	6.00	2.75
Number of Type of Devices Owned	4.14	1.52	4.00	2.00
Usage of Multiple Devices (1 - Strongly Disagree/7 - Strongly Agree)	5.73	1.39	6.00	2.00

Regarding their programming experience, participants were asked to answer some questions to assess how experienced they were in general, what programming languages were they already familiar with, and what was their level of experience with *JavaScript*, since it was the programming language that they used as part of our study. Only one participant rated their experience with *JavaScript* as 1 out of 7 because they had no previous experience with this particular language (see Table 6.8 and Figure 6.9). Everyone else had some experience with *JavaScript* and all of the participants also had experience with other programming languages (see Figure 6.10).

We were also interested in knowing which type of devices they owned, and if they tended to use more than one device at a time, since this could be an indicator of how

Table 6.8: Self-assessment of the participants experience with the *JavaScript* programming language. 1 means *Beginner* and 7 means *Expert*.

	1	2	3	4	5	6	7
JavaScript Experience	4.5%	0.0%	13.6%	27.3%	31.8%	18.2%	4.5%

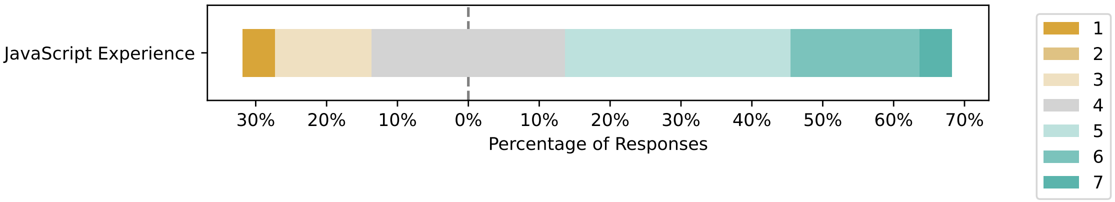


Figure 6.9: Self-assessment of the participants experience with the *JavaScript* programming language. 1 means *Beginner* and 7 means *Expert*.

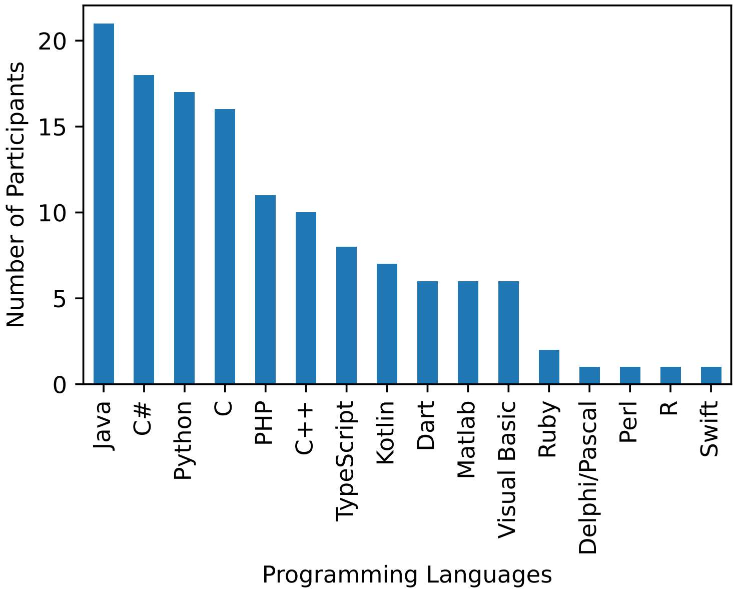


Figure 6.10: Programming languages known by the participants (other than *JavaScript*)

interesting and useful the integration of co-located devices would be for them and for other potential users. Figure 6.11 shows how many users own each of a series of common digital device types and the statistic already presented in Table 6.7 indicate that most participants own 4 or more devices. Therefore, they may already have some predisposition towards integrating multiple devices in their daily lives.

When asked if they agree with statement “I often use two or more devices at the same time in my daily life” on a scale where 1 means *Strongly Disagree* and 7 means *Strongly Agree*. We obtained the results summarized in Tables 6.7 and 6.9, and in Figure 6.12. Around 86% of the participants responded with a positive value of 5 or higher, indicating that the vast majority already engages in cross-device activities to some extent.

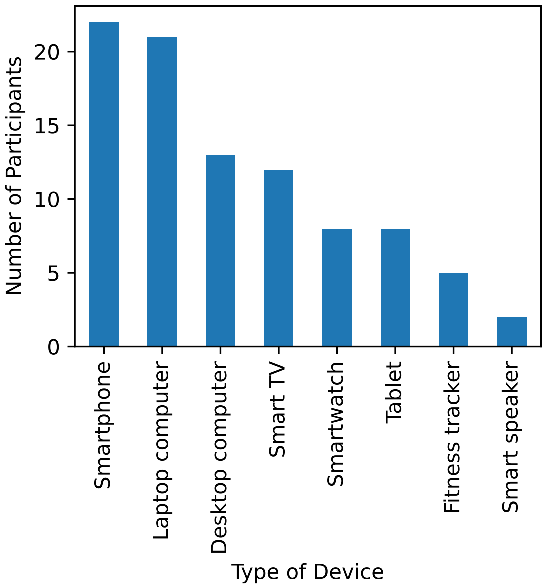


Figure 6.11: Device types owned by the participants

Table 6.9: Usage of multiple devices by the participants. 1 means *Strongly Disagree* and 7 means *Strongly Agree*.

	1	**2**	**3**	**4**	**5**	**6**	**7**
Usage of Multiple Devices	0.0%	4.5%	4.5%	4.5%	27.3%	18.2%	40.9%

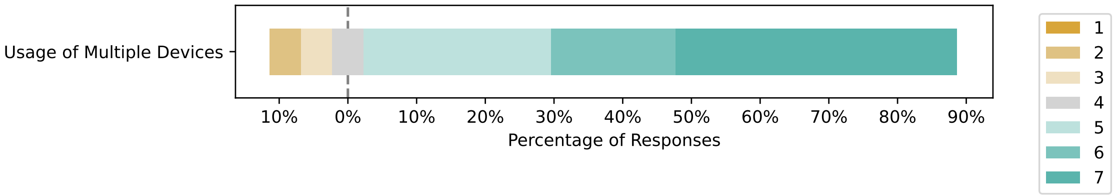


Figure 6.12: Usage of multiple devices by the participants. 1 means *Strongly Disagree* and 7 means *Strongly Agree*.

6.4.2 User Study Sessions

Each participant went through a testing session that took around 45 min to 1 h. Each session was comprised of an initial briefing that introduced the participants to concept of applications that span their interface across multiple co-located devices using by using the YanuX Skeletron as an example. They were shown how the two components of the UI were automatically distributed across the two devices of our test setup (the *PC* and *Smartphone 1* from Table 6.1). In this case, the smartphone would display the *Form* component and the *PC* would show the *Display* component. It was then demonstrated how the devices would transition to showing both components when they were no longer associated. This could be due to users holding *Smartphone 1* with their backs turned away from the *PC*, or because they moved several meters away from the *PC*.

The components of the *YanuX Framework* were briefly explained based on the architecture diagram Figure 3.1. The YanuX Coordinator was the main focus of this explanation because it is the library that developers have to use to integrate their applications with the rest of the framework. They were also introduced to the *YanuX Resource Management Element* and to the *YanuX Component Distribution Element* which are part of the YanuX Coordinator and that are integrated into the YanuX Skeletron example application.

The requirements that were defined for the UI components of the application using our *JSON-based DSL* were presented to the participants so that they could understand how the automatic distribution of components across co-located devices was decided. The base configuration was the one presented in Listing 5.4, but we also demonstrated how some changes to the definition of the requirements would affect the distribution in different scenarios.

After the initial briefing and demonstration, we replaced the code of the YanuX Skeletron application with an incomplete version that participants needed to complete in order to make the application work as in the initial demonstration. There were a total of 11 exercises that required participants to either use the API and components offered by the YanuX Coordinator, or to understand how they could possibly structure their application in a way that it can be integrated with the *YanuX Framework*. The exercises that the participants were asked to complete were the following:

1. Save the current application state to the currently subscribed *resource* using the *Coordinator* API.

2. Initialize the *ComponentsRuleEngine* instance that will be used to determine the automatic distribution of the UI elements based on the proxemic relationships of co-located devices.

3. Use the *Coordinator* API to subscribe to changes in a *resource* that stores the application's current UI state by registering the *resourceSubscriptionHandler* function as a handler.

4. Implement the body of the *resourceSubscriptionHandler* function. The UI of the application should be updated with the newly received data from the subscribed *resource* using a function that is already implemented in the application code.

5. Use the *Coordinator* API to subscribe to changes in the proxemic relationships of the devices that are running the application by registering the *proxemicsSubscriptionHandler* function as a handler.

6. Implement the body of the *proxemicsSubscriptionHandler* function by calling a function that is part of the application code to update the distribution of UI components.

7. Implement the body of the function that updates the distribution of UI components by using the *updateComponentsDistribution* method from the *Coordinator* API that updates the distribution of UI components by passing the previously initialized *ComponentsRuleEngine*. The currently available application instances, the proxemic relationships of the devices, and the *JSON-based DSL* requirements are all taken into consideration. This method is also responsible for updating the *YanuX Components Distribution Element* according to the distribution of UI components that was determined.

8. Add an event listener to the *YanuX Resource Management Element* that listens for the *resource-selected* event that is triggered when a user selects a *resource* from the drop-down list.

9. Implement the body of the function that gets called when a user selects a *resource* in the *YanuX Resource Management Element* by using a method from the *Coordinator* API to select a new *resource* based on the information contained in the event.

10. Add an event listener to the *YanuX Components Distribution Element* that listens for the *updated-components-distribution* event that is triggered when users click on the checkboxes to change the distribution of the UI.

11. Implement the body of the function that gets called when users click on the checkboxes of the *YanuX Resource Management Element* by using a method from the *Coordinator* API that distributes the components based on the information contained in the event.

The aforementioned *PC*, where one of the application instances was running in the *Google Chrome* browser, was also used to write the code needed to complete the exercises mentioned above. This computer was running the *Kubuntu 21.04 Linux* distribution which uses the *KDE Plasma* desktop environment. *Visual Studio Code* was the IDE used to edit the *main.js* file in order to complete the exercises. This allowed participants to explore the API of the various components provided of the YanuX Coordinator library by using the intelligent code completion features of the IDE. Participants also had access to the full documentation to the *TypeDoc* documentation of the library available at: `https://yanux-framework.github.io/YanuX-Coordinator/`.

Each exercise was carefully explained to the participants before they tried to solve it. Instructions were repeated or clarified when participants asked for it. They were free to explore while they tried to find a solution for the exercise. However, they were offered help if at any point they became unable to continue, or if they started to make errors that were moving them away from the correct solution. Therefore, every participant was eventually able to arrive at the solution, albeit with varying degrees of assistance. Participants were asked to rate between 1 to 7 how complicated they found the solution to be given what they were trying to achieve, where 1 stands for *Very Complicated* and 7 means *Very Straightforward* (see Appendix D).

Participants were also asked to answer to a questionnaire after they completed the 11 exercises (see Appendix E). The questionnaire started by asking several questions about the demographics and the background of the participants. Afterwards, they were asked about their level of agreement with several domain specific statements and with the two *UMUX-Lite* items [LUM13; LUM15], where 1 meant *Strongly Disagree* and 7 meant *Strongly Agree*. The questionnaire also includes a semantic differential scale of items with several opposing terms to better understand the opinion that the participants have about the proposed framework. Finally, participants were asked if they would recommend the framework to other developers (a variation of the NPS question [Rei03]) and were told to fill out an unweighted NASA-TLX form in order to assess the workload that they went through was they completed the exercises [Har06; HS88; NAS21]. Additionally details about the questionnaire, the obtained results and their interpretation can be found in the next section.

6.4.3 Results and Discussion

We now present and discuss the results for the various questions answered by our participants at the end of the user study session.

6.4.3.1 Domains Specific Questions

We now present the results for the various questions answered by our participants that aim to evaluate their experience during the development of an example application using the *YanuX Framework*. Table 6.10 and Figure 6.13 present the results of the domain specific questions ($DS1$ to $DS8$) and the *UMUX-Lite* items ($UL1$ and $UL2$) [LUM13; LUM15]. In both cases, participants were instructed to rate their level of agreement with each statement on a scale of 1 (*Strongly Disagree*) to 7 (*Strongly Agree*).

Regarding $DS1$, the vast majority of the participants agreed that it made sense to encapsulate the UI state of an application into an object. This gives us some confidence that this was a good choice on how to abstract and synchronize the UI across multiple application instances running on multiple devices. The results of $DS2$ also indicate that the vast majority of the participants understood how to save the object representing the UI state whenever there is a change in an instance to propagate the changes to the other instances, which was the task they performed as part of the *Exercise* 1. Moreover, the answers to $DS3$ also confirm that participants got a good understanding on how to use the *Coordinator* API to perform the various tasks that were part of the proposed exercises.

$DS4$ focused on assessing if participants understood how to use the *JSON-based DSL* to define the requirements of each UI component in order for them to be automatically distributed according to the capabilities of co-located devices. The vast majority of the participants rated this statement with a 6 or a 5. These results are slightly less positive

Table 6.10: *YanuX Skeletron* User Study: Statistics summary of domain specific and *UMUX-Lite* questions

1	2	3	4	5	6	7	$\tilde{x}$	$\bar{x}$	s
DS1: It makes sense that the UI state of an application is represented by an object.									
0.0%	0.0%	0.0%	4.5%	9.1%	27.3%	59.1%	7.00	6.41	0.85
DS2: I understood how to save the UI state of an application every time it changes.									
0.0%	9.1%	0.0%	0.0%	0.0%	50.0%	40.9%	6.0	6.05	1.40
DS3: I understood how to use the *Coordinator* API to perform various tasks that make my cross-device application behave properly?									
0.0%	0.0%	0.0%	0.0%	13.6%	54.5%	31.8%	6.0	6.18	0.66
DS4: I understood how the Domain-specific Language (DSL) for the automatic distribution of UI components works.									
0.0%	0.0%	9.1%	0.0%	36.4%	40.9%	13.6%	6.0	5.50	1.06
DS5: I understood how to use the *ComponentsRuleEngine* to determine the appropriate distribution of UI components based on the restrictions placed on them and the proxemics relationships between devices currently running application instances.									
0.0%	0.0%	0.0%	9.1%	40.9%	31.8%	18.2%	5.5	5.59	0.91
DS6: I understood how the custom *YanuX Resource Management Element* can be used to manage multiple application states of an application.									
0.0%	4.5%	0.0%	0.0%	9.1%	59.1%	27.3%	6.0	6.00	1.07
DS7: I understood how the custom *YanuX Components Distribution Element* can be used to manage the distribution of UI components of an application.									
0.0%	0.0%	0.0%	4.5%	13.6%	59.1%	22.7%	6.0	6.00	0.76
DS8: It makes sense to distribute the UI components of an application depending on how closely related the surrounding devices are.									
0.0%	0.0%	0.0%	0.0%	4.5%	36.4%	59.1%	7.0	6.55	0.60
ULI1: The *YanuX Framework*'s capabilities meet my requirements.									
0.0%	0.0%	0.0%	9.1%	22.7%	31.8%	36.4%	6.0	5.95	1.00
ULI2: The *YanuX Framework* is easy to use.									
0.0%	0.0%	0.0%	9.1%	31.8%	31.8%	27.3%	6.0	5.77	0.97

than the previous questions. This may due to the fact that the DSL was only explained and demonstrated during the briefing at the beginning of the session. Other than that, the provided definition was only part of the parameters passed to the *ComponentsRuleEngine* initialization in *Exercise* 2. Therefore, this may be one of the reasons why participants were not as confident about how the DSL can be used. Nevertheless, this is a point that can also be surely improved in the future by providing better documentation. In fact, *DS*5 focuses primarily on the usage of the *ComponentsRuleEngine* (covered by *Exercises* 2 and 7) and also has similar results. Most participants selected 5 or 6 as an answer showing that there is some room for improvement when it comes to better explaining to developers how to deal with the automatic distribution of UI elements across devices.

When it comes to *DS*6 and *DS*7, the results were similar and very positive. The vast majority of participants had a high degree of understanding about the *YanuX Resource Management Element* and the *YanuX Components Distribution Element*, i.e, the two reusable elements provided by YanuX Coordinator. Some of the aspects of the usage of these

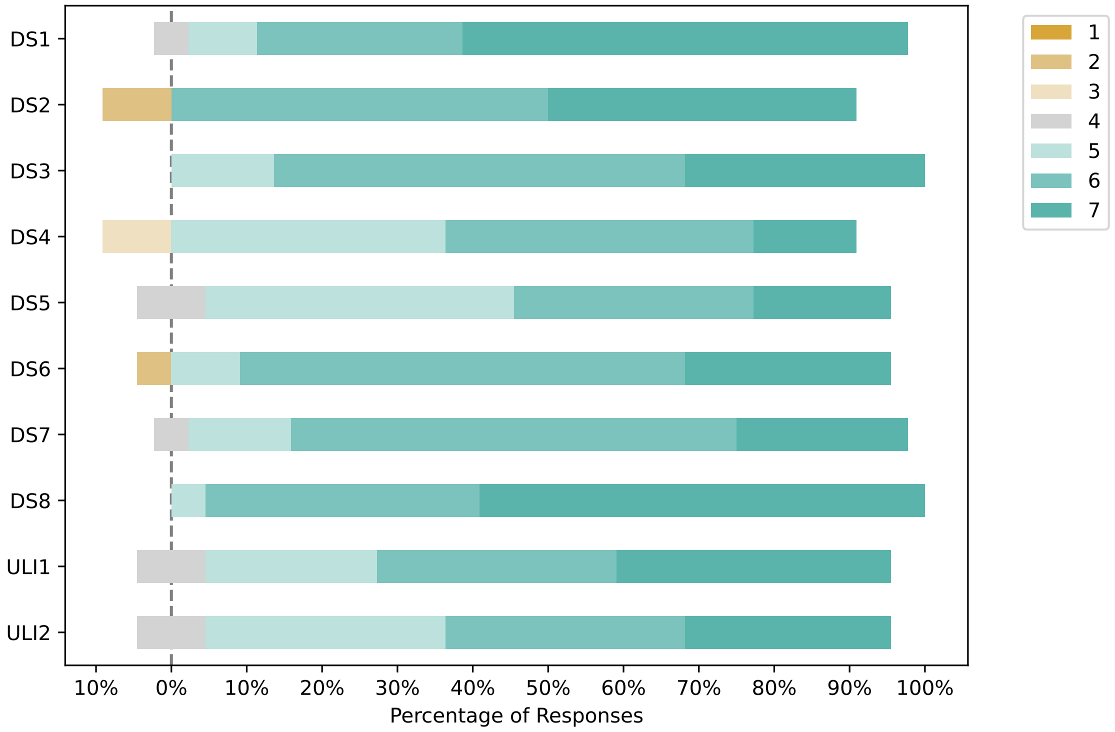


Figure 6.13: *YanuX Skeletron* User Study: Response distribution of domain Specific and *UMUX-Lite* questions

components were covered by *Exercises* 8 to 11. They were also explained during the briefing at the beginning of each session, namely the fact that multiple application states (i.e., *resources*) can be manipulated using the *YanuX Resource Management Element* and that the distribution of components can be controlled by users using the *YanuX Components Distribution Element*. Therefore, it is not unexpected that the participants got a good a understanding about these concepts. Moreover, we can expect that other potential developers are also able to grasp these concepts as long as they have access to clear documentation.

Finally, the results of $DS8$ strongly suggest that our participants agree that it makes sense to distribute the UI components of applications based on how closely related are the devices present in the surrounding environment. Just like in previous studies, this confirms that the development of cross-device applications that spread their UI components across co-located devices is an objective worth pursuing.

Regarding $ULI1$ and $ULI2$, these two questions correspond the two items that are part of a standard *UMUX-Lite* questionnaire [LUM13; LUM15]. $UL1$ addresses with the usefulness of the *YanuX Framework* and $UL2$ focus on its ease of use. In both cases the results were very positive with $UL1$ having slightly better results. This means that in general participants considered the tools provided by the framework to build cross-device applications as being useful and valuable. In the case of $UL2$, the results seems to indicate that participants did not find the framework overly complex to use. There is only so much that can be done to ease developers into the framework and the concept of cross-device applications. However, there may be still room for improvement in the design of some aspects of the framework and in providing better documentation.

Figure 6.14 presents the *Kendall's tau-b* (τ_b) correlation coefficients between the answers to domain specific and *UMUX-Lite* questions. The cells with a with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$, thus being statistical significant. Looking at those marked cells, we can see that there is a strong positive correlation between the answers to the $DS2$ and $DS3$. Both questions address operations that are done with the *Coordinator* API, so it is natural that the answers are correlated.

The strong positive correlations between $DS3$ and $DS5$, $DS3$ and $DS6$, and $DS3$ and $DS7$, also suggest that there is a relationship between the answers of the participants regarding the overall ease of use of the *Coordinator* API ($DS3$) and their understanding about other aspects, such as the usage of the *ComponentsRuleEngine* ($DS5$), of the *YanuX Resource Management Element* ($DS6$) and of the *YanuX Components Rule Engine* ($DS7$). Similar reasons should be behind the strong positive correlations between $DS5$ and $DS6$, $DS5$ and $DS7$, $DS6$ and $DS7$.

Finally, there is also a strong positive correlation between the answers to the overall ease of use of the framework ($ULI2$) and both the understanding of the *Coordinator* API ($DS3$) and the DSL to define the requirements of UI components ($DS4$). This suggests that participants who considered that the framework was easy to use were also capable to understand those concepts well, and vice-versa.

6.4.3.2 Semantic Differential Scale

Table 6.11 presents a summary of the answers to a series of 6 items of a *Semantic Differential Scale*. The distribution of the answers is also presented graphically in Figure 6.15. Each item was rated between two opposite terms with 1 representing one end and 7 the other.

In the case of the $SDS1$ item, the vast majority of participants considered the *YanuX Framework* to be *interesting* since they rated the item with a 6 or 7. Therefore, this is a clear indicator that our concept interested them and that they may want to explore our framework and the concept of applications that spread their UI across co-located devices.

Regarding the $SDS2$ item, we can see that most answers were spread out between 4, 5 and 6 which indicates that generally participants considered the framework to be either neutral in terms of complexity or at least considered it to be relatively *simple*. In fact, only 2 participants answered with a 2, meaning that they considered the framework to be *complex*. However, there were also 2 participants that considered it to be very *simple*. Therefore, this shows that, while there is room to improve the apparent complexity of

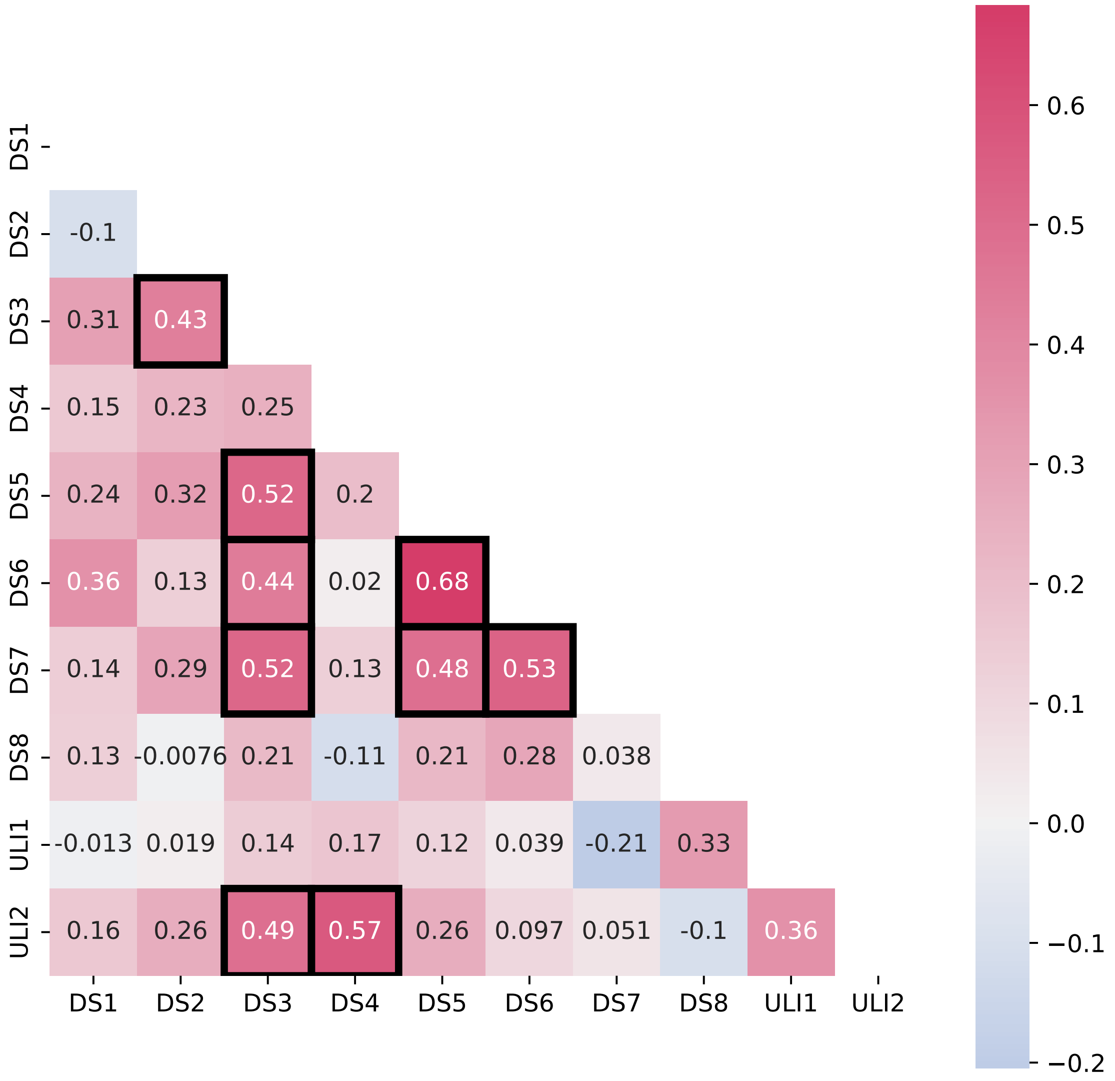


Figure 6.14: *YanuX Skeletron* User Study: *Kendall's tau-b* (τ_b) correlation coefficients between the answers to domain specific and *UMUX-Lite* questions. Cells with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$.

our framework for developers that get to use it for the first time, most participants did not consider it to be too overwhelming.

The objective of the *SDS3* item was to understand whether participants considered that the abstractions and tools provided by the *YanuX Framework* were *appropriate* or *inappropriate* for the development of cross-device applications. The response were mostly concentrated on the *appropriate* side with almost all participants answering 6 or 7. Therefore, this indicated that the framework answers appropriately to the needs of the developers when building this type of applications.

When it comes to the *SDS4* item, the objective was to assess how *easy* or *hard* was to understand the concepts behind the framework and how it works. The majority of

Table 6.11: *YanuX Skeletron* User Study: Summary of the statistics for the *Semantic Differential Scale* items.

1	**2**	**3**	**4**	**5**	**6**	**7**	$\tilde{x}$	$\bar{x}$	s
SDS1: 1 - Uninteresting / 7 - Interesting									
0.0%	0.0%	0.0%	0.0%	4.5%	45.5%	50.0%	6.5	6.45	0.60
SDS2: 1 - Complex / 7 - Simple									
0.0%	9.1%	0.0%	27.3%	27.3%	27.3%	9.1%	5.0	4.91	1.34
SDS3: 1 - Inappropriate / 7 - Appropriate									
0.0%	0.0%	0.0%	0.0%	13.6%	40.9%	45.5%	6.0	6.32	0.72
SDS4: 1 - Hard to Understand / 7 - Easy to Understand									
0.0%	0.0%	13.6%	13.6%	31.8%	22.7%	18.2%	5.0	5.18	1.30
SDS5: 1 - Hard to Learn / 7 - Easy to Learn									
0.0%	0.0%	9.1%	9.1%	18.2%	36.4%	27.3%	6.0	5.64	1.26
SDS6: 1 - Hard to Develop /7 - Easy to Develop									
0.0%	4.5%	4.5%	9.1%	31.8%	31.8%	18.2%	5.5	5.36	1.29

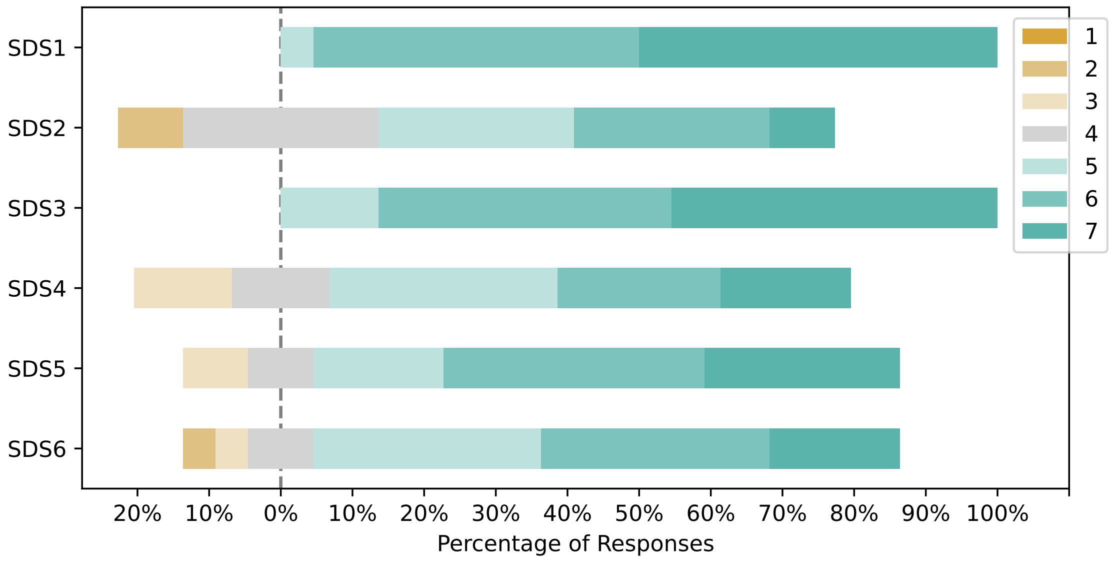


Figure 6.15: *YanuX Skeletron* User Study: Response distribution of the *Semantic Differential Scale* items.

the participants answered with a 5 or 6, and the third most chosen choice was 7. All of these lean towards the *Easy to Understand* side, suggesting that most participants did not find our framework overwhelmingly difficult to understand. However, there is still room for improvement when it comes to make it more understandable, especially considering that 27.2% participants answered with a 4 (*neutral*) or even a 3 (leaning towards *Hard to Understand*).

The *SDS*5 item focused on the learnability of the framework. The results were better than in the previous item, with the most chosen choices being 6 and 7, which lean heavily towards the *Easy to Learn* side. There were also less participants that answered with 4 (*neutral*) or even 3 (leaning towards *Hard to Learn*). Similarly, the $SDS6$ focuses on the ease of developing applications using the framework, with the majority of the participants choosing 5 or 6 (tied in first place). The second most chosen answer was 7. All of these values lean more towards the *Easy to Develop* spectrum. Nevertheless, there were still some participants that chose 4 (*neutral*) or even 2 or 3 (leaning more towards the *Hard to Develop* side).

These results lead us to believe that in general the APIs and components provided by the YanuX Coordinator are structured in a way that is relatively easy to learn and use by potential developers. However, there is still room for improvement and some aspects that should be addressed. In fact, there is some concrete feedback that we took note as participants went through the 11 proposed exercises that we intend to incorporate into newer versions of the framework.

It is also interesting to analyze the *Kendall's tau-b* (τ_b) correlation coefficients between the answers to the *Semantic Differential Scale* items in Figure 6.16. Please note that the cells with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$, thus being statistical significant.

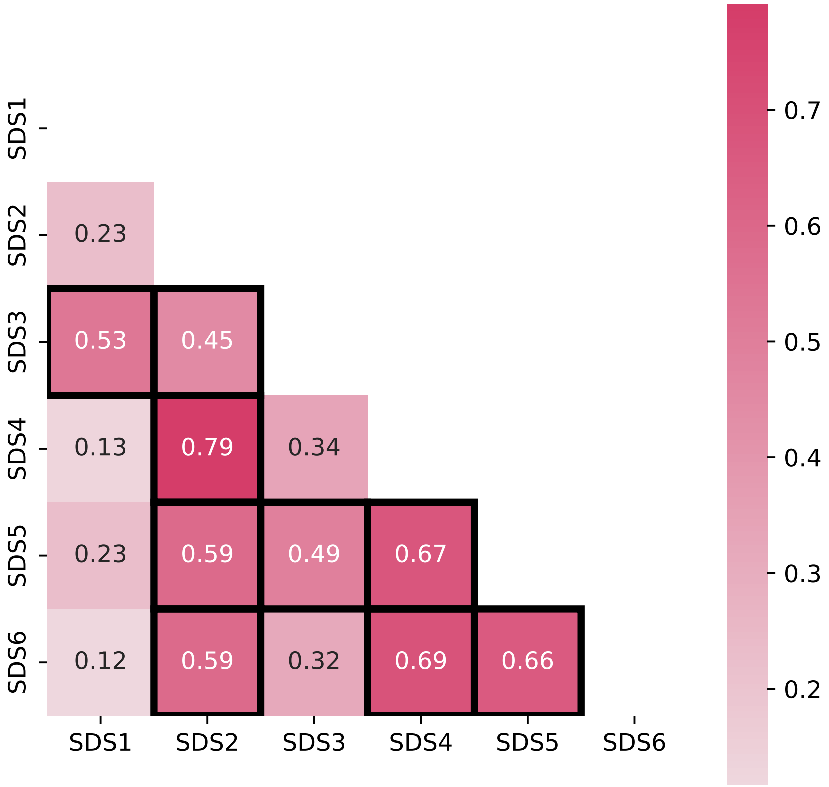


Figure 6.16: *YanuX Skeletron* User Study: *Kendall's tau-b* (τ_b) correlation coefficients between the answers to the *Semantic Differential Scale* items. Cells with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$.

$SDS2$ and $SD4$ have a very strong positive correlation, suggesting that participants that considered the framework *simpler* also considered it *easier to understand*, and vice-versa. The $SDS2$ item is also correlated with both $SDS5$ and $SDS6$ items, suggesting a relationship between *complexity* and the difficulty to *learn* or to *develop* applications using the framework. Additionally, $SDS2$ has a considerable correlation with $SDS3$, which suggests that the participants which considered the framework to be *simpler* also tended to consider it more *appropriate*, and vice-versa.

The correlation between $SDS1$ and $SDS3$ suggests that there is a relationship between considering the framework more or less *interesting* and more or less *appropriate*. While the significant correlation between $SDS3$ and $SDS5$ suggests a similar relationship between the *appropriateness* of the framework and *learnability*. Finally, $SDS4$ is correlated with both $SDS5$ and $SDS6$, and $SDS5$ is also correlated with $SDS6$. This is not surprising since these items are related to some form of *ease of use* which may represent a latent variable.

6.4.3.3 Post-Exercise Questions

Participants were asked asked to rate how difficult or complicated they found the solution to be for what they were trying to achieve after each of the 11 exercises. In this case, 1 stood for *Very Complicated* and 7 meant *Very Straightforward*. Table 6.12 and Figure 6.17 present a summary of the participants' responses.

Table 6.12: *YanuX Skeletron* User Study: Summary of the statistics for the post-exercise question.

Ex.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	$\tilde{x}$	$\bar{x}$	μ
1	0.0%	0.0%	4.5%	13.6%	13.6%	54.5%	13.6%	6.0	5.59	1.05
2	0.0%	4.5%	0.0%	18.2%	45.5%	31.8%	0.0%	5.0	5.00	0.98
3	0.0%	0.0%	9.1%	4.5%	0.0%	45.5%	40.9%	6.0	6.05	1.21
4	0.0%	0.0%	0.0%	0.0%	4.5%	9.1%	86.4%	7.0	6.82	0.50
5	0.0%	0.0%	0.0%	0.0%	0.0%	27.3%	72.7%	7.0	6.73	0.46
6	0.0%	0.0%	0.0%	0.0%	9.1%	18.2%	72.7%	7.0	6.64	0.66
7	0.0%	0.0%	0.0%	4.5%	22.7%	50.0%	22.7%	6.0	5.91	0.81
8	0.0%	0.0%	0.0%	4.5%	9.1%	9.1%	77.3%	7.0	6.59	0.85
9	4.5%	4.5%	0.0%	9.1%	27.3%	45.5%	9.1%	6.0	5.23	1.45
10	0.0%	0.0%	0.0%	4.5%	4.5%	18.2%	72.7%	7.0	6.59	0.80
11	0.0%	0.0%	0.0%	0.0%	9.1%	22.7%	68.2%	7.0	6.59	0.67

As previously mentioned, participants had to find out how to use the *Coordinator* to update the currently selected *resource* with the most recent application state as part of exercise 1. They were still not familiarized with some of the conventions and nomenclature of the API so they spent some time exploring the auto complete suggestions provided by *Visual Studio Code* or the provided documentation (i.e., `https://yanux-framework.github.io/YanuX-Coordinator/`). Some participants had difficulty in discovering that the right method was *setResourceData* since they started looking for a method with *update* in their name. In fact, there is a method called *updateResources*

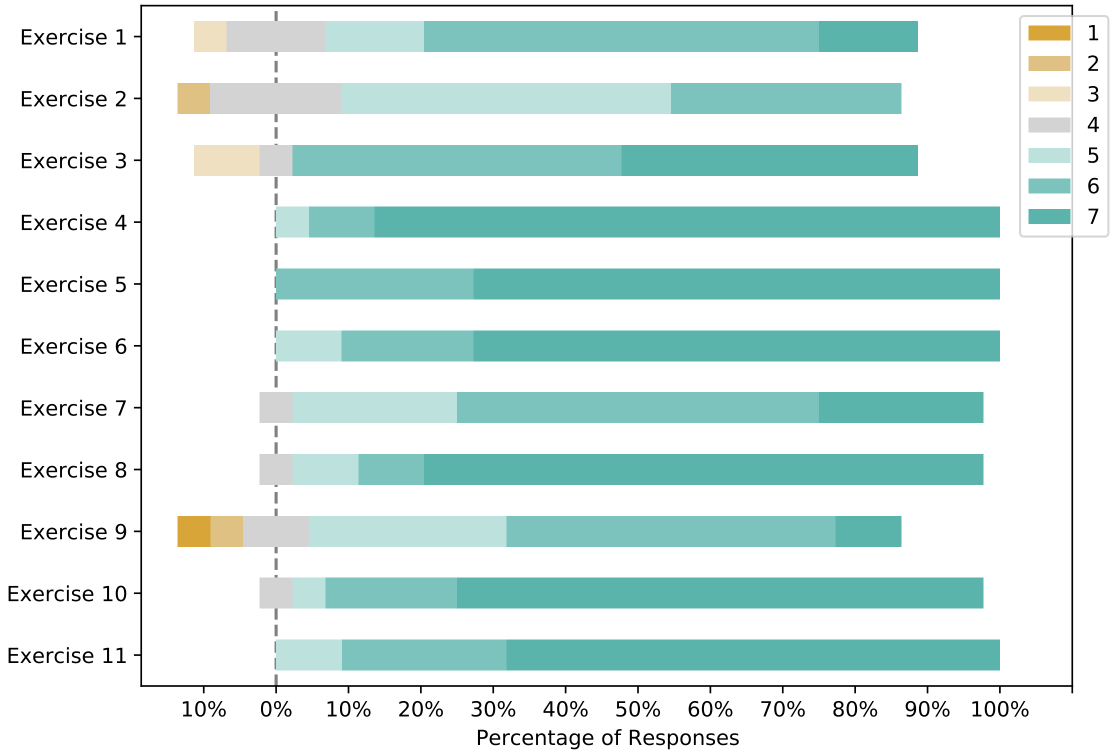


Figure 6.17: *YanuX Skeletron* User Study: Response distribution for the post-exercise question.

but it has a different purpose. Others assumed that they had to retrieve some object representing the current *resource* and use it to update it on the server side. Either way, with some hints the participants were able to find the *setResourceData* method and use effectively.

Despite some difficulties and the fact that this was the first contact that participants had with the framework, most of the participants gave exercise 1 a rating of 6, with some choosing 7. Nevertheless, 31.1% of the participants chose 5 or less. Moreover, the median and mean of the responses were 6.0 and 5.59, respectively. These were the third lowest values out of the 11 exercises. Therefore, despite the novelty factor, we believe that there may still some room for improvement. For instance, changing the name of the method or providing an alternative one called *updateResource* or *updateResourceData*, which seems to align better with what potential developers expect to find.

On exercise 2 participants had to initialize a *ComponentsRuleEngine* object in order for it to later be used to automatically distribute the UI components based on the restrictions previously presented in Listing 5.4, the proxemic relationships between co-located devices and the information about the active instances of the application that are running on those devices. Therefore, they had to use the *ComponentsRuleEngine* constructor and pass three parameters: the UUID of the currently running instance, the UUID of the currently running device, and the restrictions for the UI components that were previously defined and explained to the participants as part of the initial briefing.

Some participants had some difficulty understanding that they needed to create a new instance of the class *ComponentsRuleEngine* but most understood that right away. Nevertheless, some had to be told to use the keyword *new* because they did not know or remember how to call a constructor in *JavaScript*. The next step was passing the three parameters to the constructor. Participants had no problem with the last parameter, other than recalling the name of the variable that was shown to them during the briefing. The first two parameters were a little bit more problematic. The first reaction of most participants was to find existing variables that they could use. However, there were no variables directly available with the required information. Therefore, most participants had to receive a hint that they could use the variable containing an initialized *Coordinator* instance and explore the members of that class for the required values. Multiple participants had a tendency of focusing on *methods* rather than properties, so in those cases they were hinted that there also *properties* available and that the answer could possibly be found there. That usually lead them to finding the *instance* and *device* which are the properties that store information about the current *instance* and *device* that are associated with the current *Coordinator* object.

In some cases, participants tried to pass the properties they found directly as parameters and had to eventually be told that they were going in the right direction but that they still needed to find the correct information by looking at the properties that the *instance* and *device* had available. There were some participants that did not need those hints and started looking at the properties right away. In either case, only a few were able to find the right property at the first try. Most of the participants ended selecting the *_id* or *id* properties which are not UUIDs as required by the constructor. Those properties are just the database identifiers (*_id* is used by default and *id* is just a copy of it). Therefore, they had to be told that they were looking for an UUID and then they were able to find the *instanceUUID* and the *deviceUUID* properties that were part of the *instance* and *device*, respectively.

Given that exercise 2 was the one that needed the most indirect access to information, it is no surprise that it was the one that got the lowest average rating with a mean and median of 5.0. In fact, it was also the only exercise in which the most common rating was below 6. Nevertheless, 5 is still on the positive side of the scale. Moreover, as this was only the second exercise we noticed that most participants were still getting used to the framework. Despite that, we feel that we could make some improvements. For instance, the *instanceUUID* and *deviceUUID* properties could simply be named UUID since they are already associated with an *instance* or *device*. It could also be possible to allow the *ComponentsRuleEngine* to receive the whole *instance* and *device* objects instead of just their UUIDs. The most extreme solution would be to simply pass the whole *Coordinator* object, or for it to optionally receive the restrictions of the UI components when it is instantiated allowing the *ComponentsRuleEngine* to be managed internally by the *Coordinator* itself.

On exercise 3 participants had to discover the appropriate *Coordinator* method to register a function that handles events coming from the server (i.e., the YanuX Broker) with the updated state of the currently subscribed resource. There were some examples already present in the source code of the subscription to other events that are supported by the *Coordinator* so that participants could understand that there was a pattern they could follow. Therefore, the vast majority of the participants could understand that they had to find a method that started with *subscribe* and that had something to do with *resource*. However, some participants got confused because there are three methods that fit that criteria: *subscribeResource*, *subscribeResources* and *subscribeResourceSubscription*.

Participants had to be made aware that *subscribeResources* and *subscribeResourceSubscription* were already being used. The former was used to subscribe to changes in the whole list of *resources* that the current user has access to, while the latter was used to subscribe to changes in the *id* of the subscribed *resource* when users select to use a different one (e.g., by using the *YanuX Resource Management Element*). Thus, The only one remaining, i.e., *subscribeResource*, would be the correct one. The name of the function that was supposed to be used as the subscriber function was already provided to the participants, so they had no difficulty in completing the exercise once they knew which one was the correct method.

The vast majority of the participants gave exercise 3 a rating of 6 or 7. Despite the difficulties, they told us that the organization of the *Coordinator* API made sense as the reasoning behind their rating. Nevertheless, there were still some participants that were not as understanding and gave a rating of 4 or 2 corresponding to 18.2% and 4.5% of the responses, respectively. Therefore, given these results and the difficulties that we noticed during this exercise, we believe that we could improve the *Coordinator* API by keeping the *subscribeResource* name but possibly changing the *subscribeResources* and *subscribeResourceSubscription* to something more distinct and descriptive to avoid confusion.

Exercise 4 is the direct continuation of exercise 3. Participants were told to complete the *resourceSubscriptionHandler* function, which was the function registered as the subscriber function for the *subscribeResource* method used in exercise 3. They were told about the parameters received by the function and that the one they should be more interested in is the *data* parameter since it holds the most up-to-date state of the subscribed *resource* when the function is called by the *Coordinator* in response to a an event received from the YanuX Broker. They were then supposed to use the *updateBmi* function, which was already part of the application code base, to update the whole application UI to reflect the most up-to-date application state.

The vast majority of the participants found exercise 4 very easy since they just needed to call the function and pass *data* as a parameter. This is reflected by the 86.4% of the participants choosing a rating of 7, 9.1% choosing 6, and only one participant (4.5%) choosing 5. The objective of this exercise was for participants to understand how to implement the subscriber function passed to the *subscribeResource Coordinator* method and how they could organize the code base of their applications in order to be able to update the UI based on an object containing the most up-to-date application state. From the previously mentioned results, we can conclude that this approach is considered adequate by the vast majority of the participants and that no direct changes are required to this part of the framework.

Exercise 5 is similar to exercise 3 in that participants also have to subscribe to a type of event using the *Coordinator* API. In this case, they need to register a function in order to subscribe to the changes in the proxemic relationships of the devices that are currently running the application. Similarly to exercise 3, participants started by exploring the methods that are available to them and they had little trouble in finding the correct method named *subscribeProxemics*. This time there were no other methods with similar names that could confuse them. Besides that, the fact that exercise 5 is similar to a previous one may contribute towards a feeling of familiarity. Therefore, the results were mostly very positive with the vast majority of the participants (72.7%) rating the exercise with a 7 and the remaining 27.3% rating it with a 6. This means that this is also an area of the *Coordinator* API that does not need a redesign.

Regarding exercise 6, it is similar to exercise 4 in that participants have to complete the body of the handler function registered using the *subscribeProxemics* method. In this case, we explained what were the parameters received by the function and that participants should use the *updateComponentsDistribution* function to update the distribution of the UI components based on the changes that happened to the proxemic relationship of the devices. In general, participants had little trouble calling the function and were only surprised by the fact that it received no parameters.

It was explained that this pattern allowed the *updateComponentsDistribution* function to be reused across different situations. For instance, it was used when both *proxemics* and *instances* change. However, the information available in one case might not be available in another, i.e., when *proxemics* relationships change we have information about the overall *proxemics* but when an *instance* changes we only have information about the *instance* that change. Therefore, the internal implementation of the *updateComponentsDistribution* function will retrieve all the needed information from the YanuX Broker. This may be less efficient but it is easier to implement and understand. If performance ever becomes an issue it is possible to make a more efficient implementation that does not require, or that minimizes, the retrieval of information from the YanuX Broker.

In terms of the post-exercise ratings given to exercise 6, the results were also very positive and mostly in line with the previous exercise. 72.7% of the participants gave a rating of 7 and 18.2% rated it with 6. Only a small minority of 9.1$, which represents 2 partipants, chose a lower rating of 5. Therefore, we also believe that this approach does not need any significant change in future iterations of the framework.

Exercise 7 is focused on the implementation of the body of the aforementioned *updateComponentsDistribution* function. Participants were told to use a method of the *Coordinator* that is also called *updateComponentDistribution* in order to automatically distribute the UI components based on the most up-to-date information about *instances* and *proxemic* relationships. The main difficulty of this exercise is to find the correct parameters to pass to the method: a *ComponentsRuleEngine* instance, a function that gets called when the distribution is determined and an object with the *YanuX Components Distribution Element* that will display the current UI distribution.

Some participants remembered that they initialized a *ComponentsRuleEngine* as part of exercise 2 and went looking for the variable name. Others had to be remembered of that, but were also usually capable of finding the right variable. Regarding the second parameter, they had to be informed that there was already a function called *configureComponents* that they could use. This was to be expected since they were not the ones that programmed the application from scratch. The last parameter was also obvious for many participants since when we introduced the *updateComponentsDistribution* function we introduced a line that was already present and that retrieved the *YanuX Components Distribution Element* using the *getElementById* method of the DOM API.

Regarding the ratings for exercise 7, half of the participants gave it a rating of 6. The remaining half was almost perfectly split between 5 and 7, with only a participant giving a rating of 4. These are mostly positive results given that this exercise required more effort to understand and the need gather the right parameters to pass to the *updateComponentsDistribution* method. As previously mentioned in the analysis of the results for exercise 2, we could possibly simplify the usage of the *ComponentsRuleEngine* by having it be managed internally by the *Coordinator*. However, this separation of concerns was actually an initial design decision since we believe that the usage of the *ComponentsRuleEngine* should be optional. Developers should be free to use our automatic distribution algorithm or to use their own solution. By instattiating and managing the *ComponentsRuleEngine* from within the *Coordinator* we would be pushing developers harder towards using it.

Exercise 8 focused on registering a listener to deal with one of the custom events provided by the *YanuX Resource Management Element*. The listeners for all but one of the custom events were provided as examples. The only one missing was the *resource-selected* event which is fired when a user selects a *resource* from the drop-down list that shows all the *resources* that the user has access to. Participants were provided with the correct name of the event and with the name of the function that was supposed to be used as the listener. Given the examples that were provided for the other events, most participants had little trouble in registering the event using the *addEventListener* method of the *YanuX Resource Management Element* which is a standard approach for any element when using the DOM API.

The ratings given by the participants show that most of them found this method of registering events for the *YanuX Resource Management Element* to be appropriate, with 77.3% of the particpants gaving it a rating of 7. There were few participants that gave lower ratings, with the lowest being a 4 from a single participant. These results are largely positive and we believe that this is an area that does not need changes. In fact, it would be difficult to change it since this is the standard approach that is part of the DOM API. The only thing that can eventually be improved is the naming of the events and their documentation.

Exercise 9 is a continuation of exercise 8 in which participants had to implement the body of the function that was registered as the listener to the *resource-selected* event. It was explained to participants that the function received a *CustomEvent* object that has a *detail* property which contains useful information about the event. This includes a *selectedResourceId* property that indicates which is the *resource* that was selected using the *YanuX Resource Management Element*. Participants were also told that the *Coordinator* has a method that they could use to select a resource. While many participants were able to find the *selectResource* method with little effort, some had a more difficulty in finding it but when it was emphasised that they had to select a *resource* they found it almost straight away. They then almost instinctively passed the *selectedResourceId* to the method. However, they had to be told that this was only partially correct since the *selectResource* method receives two parameters: a subscriber function that is called whenever there is a change in the newly selected *resource* and the *ID* of the newly selected *resource*. This generated some confusion and they had to be told that they could just use the function that was passed to the *subscribeResource* method em exercise 3. Therefore, they looked for it in the source code, or tried to find it using the autocomplete feature, and eventually passed the correct parameter to the method.

The ratings for exercise 9 reflect the difficulty found with the first parameter of the *selectResource* method with the majority of the participants choosing 6 as the rating with 45.5% followed by 5 with 27.3%. Then there were a couple of participants that rated higher with a 7 or lower with 4. Finally, there was one participant that gave it a rating of 1 and another that gave it a rating 2. Despite this, the results are still mostly positive. However, we acknowledge that the *selectResource* method can be improved by having the *resource ID* as the first parameter and the subscriber function as an optional second parameter that by default takes the value of the function previously registered using the *subscribeResource* method. This continues to allow developers the ability to provide a different function if they need to but they can also just reuse the same function without having to explicitly pass it to the *selectResource* method.

Exercise 10 was similar to exercise 8 but, instead of being about the *YanuX Resource Management Element*, the focus was on the *YanuX Components Distribution Element*. In this case, participants had to register a listener for the *update-components-distribution* event, which is triggered when someone changes the distribution of UI components manually by clicking on the checkboxes that are part of the *YanuX Components Distribution Element*, by using the *addEventListener* method of the custom element. An example for the other event supported by the element was provided and the name of the function to register as the listener was also indicated.

Since this exercise was similar to a previous one, participants had little trouble in figuring out the correct solution. This probably led to the very positive ratings, with the vast majority of the participants (72.7%) choosing a rating of 7 and 18.2% choosing a 6. Only a couple of users selected a lower rating of 4 or 5. Therefore, this is an area which believe requires little improvement in the future. As already mentioned, the approach used to add event listeners to the custom elements is through the usage of the standard *addEventListener* method of the DOM API. So users already familiar with that approach should have no problem using our custom elements, other than learning the names of the custom events and possibly the structure of the custom data passed as a *detail* of the event.

Finally, exercise 11 was focused on the implementation of the event listener function passed in the previous exercise, i.e., the *updatedComponentsDistribution* function. This function receives a *CustomEvent* with the information about the manual distribution of components made by a user by interacting with the *YanuX Components Distribution Element*. They were then told that there was method in the *Coordinator* that could be used to update the distribution of the UI components based on the selection made by the user. Most of the participants found the *distributeComponents* method since it one of the few methods related with the distribution of components. The others are the *clearComponentsDistribution* and *updateComponentsDistribution* methods, with the latter having already been used as part of exercise 7. Some participants needed a bit more of clarification about what to look for but they were eventually able to find the correct method. They then checked what the method received and noticed that it was a *CustomEvent* and correctly deduced that they could pass the event received by the listener function to the method so that it could use the information contained in the event to update the distribution of the UI components of the application.

Regarding the ratings of this exercise, they were very positive with 68.2% of the participants choosing 7 and 22.7% choosing 6. The remaining two participants (9.1%) chose 5. This leads to believe that this approach is mostly correct and that there is not a lot of room for improvement. Perhaps the naming of the method could be improved, but in general it was not problmatic. Eventually, one could also pass the distribution of the components contained in the event directly to the *distributeComponents* method instead of passing the whole event.

Besides the analysis of the ratings of the exercixes individually, Figure 6.18 presents the *Kendall's tau-b* (τ_b) correlation coefficients between the ratings for each exercise. Please note that the cells with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$, thus being statistical significant.

The results for exercises 1 and 2 seem to have a strong prositive correlation. This may be due to the fact that these were the first two exercises and that participants were still getting used to the *Coordinator* library, leading to the participants that rated exercise 1 with a lower score also rating exercise 2 lower, and vice-versa. There is also some correlation and similaries between exercise 1 and 11 since they both call a *Coordinator* method. Despite this possible explanation, please note that strong correlations between exercises that share some similarities may at least be partially due to the fact that some participants simply have a tendency to give higher or lower scores across all exercises.

Regarding the correlation between exercise 2 and 9, it may due to the fact that both

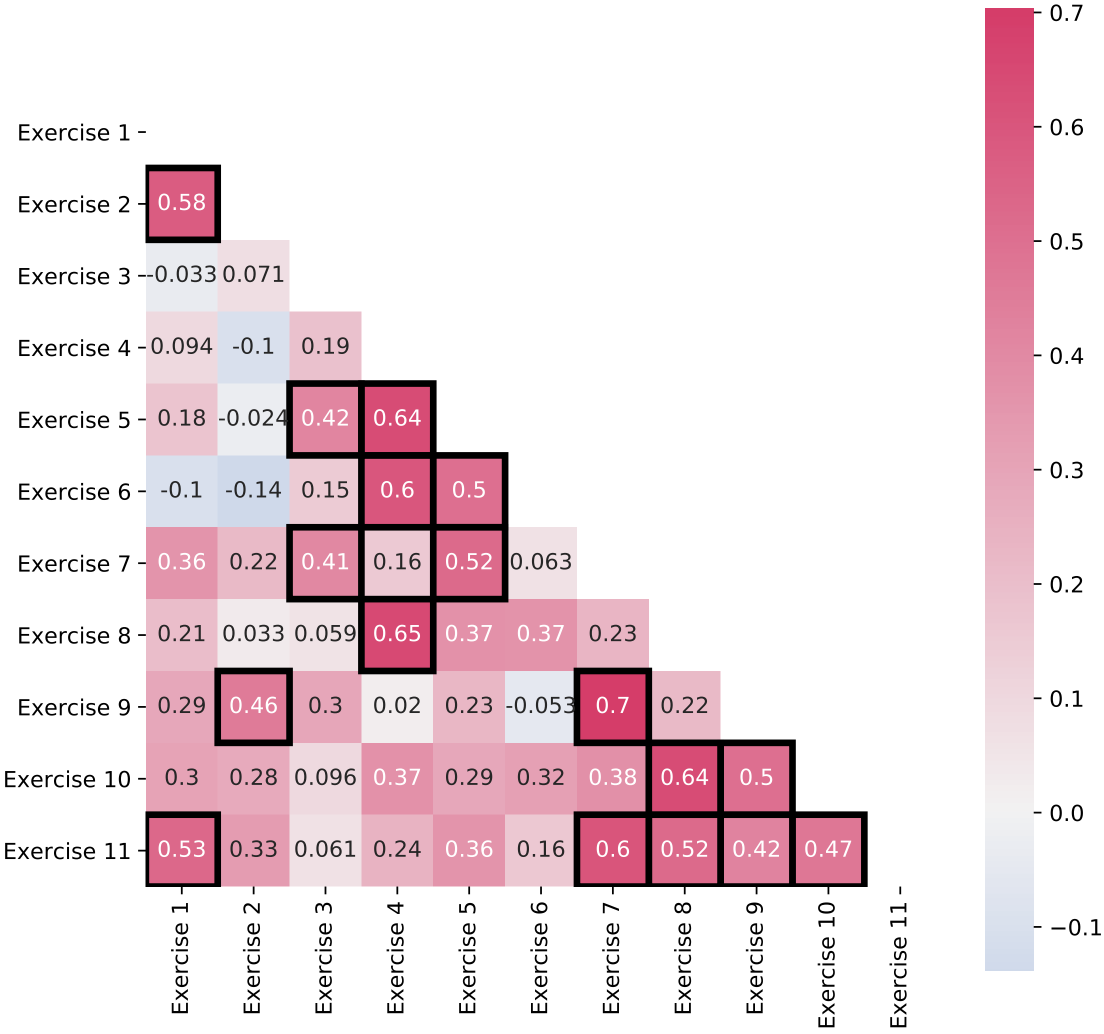


Figure 6.18: *YanuX Skeletron* User Study: *Kendall's tau-b* (τ_b) correlation coefficients between the answers to the post-exercise question items. Cells with a black border indicate that the correlation has a $p < 0.05$ for the null hypothesis that $\tau_b = 0$.

exercises required some effort from the part of the participants in determining and finding the correct parameters to pass to the *CoomponentsRuleEngine* and *Coordinator*'s *selectResource* method, respectively. Similarly, the results of exercises 3 and 5 may also be correlated because they were both focused on subscribing to events using the *Coordinator* API (i.e., using the *subscribeResource* and the *subscribeProxemics* methods, respectively). There is also a strong correlation between exercise 3 and 7 but they do not share many similarities, other than both using a *Coordinator* method.

The strong correlation between exercises 4 and 6 may due to similarities between the exercises, i.e., they call methods that are part of the application code base in response to the events that were subscribed in exercises 3 and 5, respectively. Meanwhile, exercise 5 is strongly correlated with both exercises 6 and 7. This may be explained by the fact that exercise 6 is the continuation of exercise 5, and exercise 7 is the continuation of exercise 6. However. despite the fact that exercises 3 and 4 share a similar relationship, the results of their ratings are not nearly as much correlated. Moreover, the direct correlation between exercise 6 and 7 is almost zero.

Regarding exercise 7's strong correlation with exercise 9 and 11 it may be related with the fact that the three exercises were focused on using *Coordinator* methods to perform some action: update the distribution of UI components, selecting a new resource and manually setting the distribution of UI components, respectively. Exercise 8 and 10 are also highly correlated and they are focused on the same thing, i.e., registering functions as listeners to the custom events fired by the *YanuX Resource Management Element* and the *YanuX Components Distribution Element*, respectively. Similarly, exercises 9 and 11 may be correlated since they both deal with implementing the body of the listener functions registered in exercises 8 and 10 by using the appropriate *Coordinator* methods.

Exercises 10 and 11 also display a strong correlation which may be due to the fact that the former is the continuation of the latter. Finally, there is also a strong correlation between exercises 8 and 11, and exercises 9 and 10, which may be related with the fact that exercises 8 to 11 (i.e., the last four exercises) are all related to registering listener functions to the events provided by the custom elements offered by the *Coordinator* library and implementing the body of those functions.

6.4.3.4 UMUX-Lite

It is possible to calculate an overall *UMUX-Lite* score based on the $ULI1$ and $ULI2$ items that were part of the questionnaire by using Equation 6.1 [BK16]. There is a variation of that formula called *UMUX-Liter* which is presented in 6.2 [BK16; Bor+15; LUM13; LUM15]. The second formula was adjusted so that the resulting scores have a very high correlation and correspondence with the SUS [Bro96]. However, there have been some studies that suggest that the unadjusted version can actually be better at predicting SUS values [BK16]. Despite the fact that *UMUX-Lite* and SUS are more geared for interactive systems used by end-users, these correspondences enable us to make a comparison with well known benchmark values for the SUS in order to get an overall grade for the usability of the framework [BKM09; LS18].

$$UMUX-Lite = ((ULI1-1)+(ULI2-1))*\frac{100}{12} \tag{6.1}$$

$$UMUX-Liter = 0.65*(((ULI1-1)+(ULI2-1))*\frac{100}{12})+22.9 \tag{6.2}$$

The calculated values are 81.1 for the *UMUX-Lite* and 75.6 for the *UMUX-Liter* by applying Equations 6.1 and 6.2, respectively. In the case of the *UMUX-Lite* value, it is an acceptable value which corresponds to *Good* rating and a *B* grade [BKM09]. The *UMUX-Liter* value is also acceptable and can be considered a *Good* rating, but the grade falls to a *C* [BKM09]. According to a slightly different grading scale prosed by Lewis et al., the grades would be *A* for the *UMUX-Lite* and *B* for the *UMUX-Liter*, corresponding to a percentile range of [90, 95] and [70, 79], respectively [LS18]. Either way, these SUS results are largely positive and indicate that we got many things right. Nevertheless, there is still room for improvement as part of our future work.

6.4.3.5 Net Promoter Score (NPS)

We also asked participants "How likely is that you would recommend the YanuX Framework to other developers you known?"on a scale of 1 to 10. This allow us to calculate an NPS of 31.8 for our framework [Rei03]. The answers to this question are also summarized in Table 6.13 and Figure 6.19.

Table 6.13: *YanuX Skeletron* User Study: Statistics summary of the NPS question.

1	0.0%
2	0.0%
3	0.0%
4	0.0%
5	0.0%
6	9.1%
7	27.3%
8	22.7%
9	22.7%
10	18.2%
NPS	31.8
Promoters	40.9%
Passives	50.0%
Detractors	9.1%
Median ($\bar{x}$)	8.0
Mean ($\tilde{x}$)	8.14
SD (s)	1.28

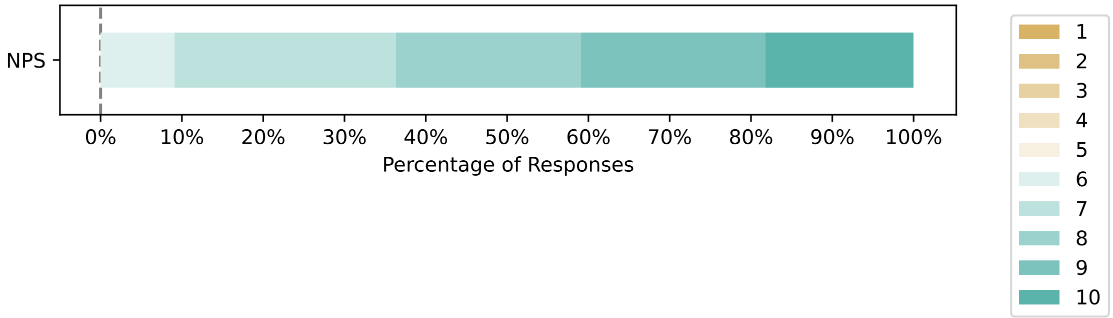


Figure 6.19: *YanuX Skeletron* User Study: Response distribution of the responses to the NPS question.

As can be seen, we had more promoters (40.9% of the participants responded with a 9 or 10) than detractors (9.1% of the participants responded with a 6 or less). The remaining half of the participants responded with a 7 or 8 which corresponds to passive stance in the calculation of the NPS. Therefore, it seems that we were able to capture the interest of a good part of the participants (*promoters*), with only a minority awarding a relatively low score (*detractors*). Even so, it was the highest score out of the ones that are often considered negative. Moreover, there is also a large amount of *passives* that can potentially be turned into *promoters* by improving some aspects of the framework, its documentation, and how we communicate and explain how it works.

6.4.3.6 NASA Task Load Index (NASA-TLX)

The final part of the questionnaire that participants responded at the end of the session consisted of the paper and pencil version of NASA-TLX that is divided into the following 6 subjective items [Har06; HS88; NAS21]:

- **Mental Demand**: How mentally demanding was the task?
- **Physical Demand**: How physically demanding was the task?
- **Temporal Demand**: How hurried or rushed was the pace of the task?
- **Performance**: How successful were you in accomplishing what you were asked to do?
- **Effort**: How hard did you have to work to accomplish your level of performance?
- **Frustration**: How insecure, discouraged, irritated, stressed, and annoyed were you?

Typically, participants are also asked to weigh each of the 15 pairs of the individual items. However, it is common to skip this step in order to keep the questionnaire shorter without a great impact on the results [Gri15; Har06]. Therefore, we used this approach and only recorded the participants' choice for each of the 6 items on a scale between 0 and 100 on 5 step intervals. A summary of the statistics of the answers of the participants to the 6 items can be found in Table 6.14 and Figure 6.20.

Table 6.14: *YanuX Skeletron* User Study: Statistics summary for the NASA-TLX.

	$\bar{x}$	s	*Min.*	Q_1	Q_2	Q_3	*Max.*
Mental Demand	45.91	22.13	10.00	30.00	47.50	65.00	75.00
Physical Demand	7.95	9.47	0.00	0.00	5.00	15.00	35.00
Temporal Demand	33.64	27.87	0.00	11.25	25.00	50.00	85.00
Performance	22.50	16.38	5.00	15.00	15.00	25.00	65.00
Effort	37.27	24.92	5.00	20.00	32.50	45.00	90.00
Frustration	13.18	15.63	0.00	5.00	10.00	15.00	65.00
Score	26.74	13.71	3.33	16.04	22.92	34.58	52.50

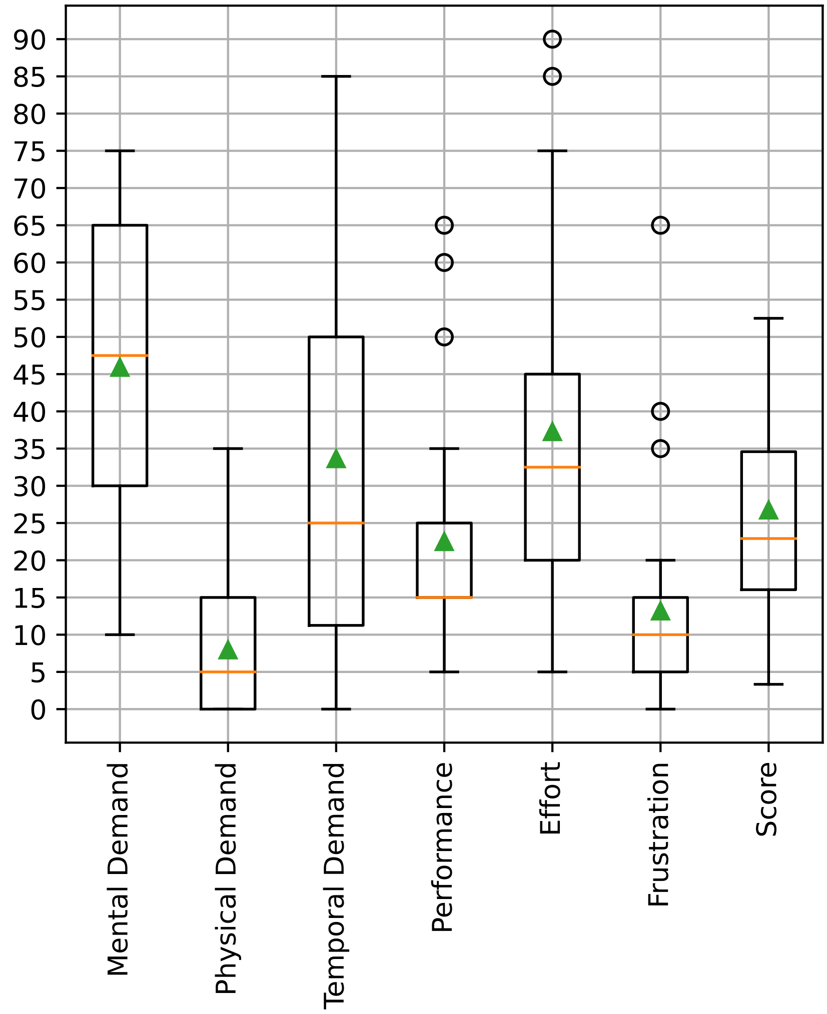


Figure 6.20: *YanuX Skeletron* User Study: Box plot of the responses to the NASA-TLX items and also the distribution of the calculated score for each participant.

There is a seventh item called *Score* which was calculated as the unweighted mean of the first 6 for each participant. The resulting mean of this item across all participants is the final resulting *Raw NASA-TLX Score* of 26.74. Ideally, given that the NASA-TLX is a measure of workload, the resulting score should not be too high because it would indicate that understanding and using the framework can be too taxing and overwhelming for potential developers. While 26.74 does not appear to be too high considering that scores can vary between 0 and 100, it is important to find a comparison reference based on previous work.

Grier presents an analysis of the results from several studies that used the NASA-TLX [Gri15]. According to that analysis, the score of 26.74 lies slightly above the 10th percentile (26.08), but well below the 20th percentile (33.00) when considering all of the reviewed studies. It is also consistently below the 25th percentile of the results grouped by different task types. The exceptions are *Card Sorting*, *Computer Activities*, *Daily Activities*, *Mechanical Tasks* and *Navigation* for which the value of 26.74 is above the 25th percentile. These type of activities are probably inherently less taxing leading to lower scores in general that cause lower values to appear across all percentiles. Nevertheless, the workload value for the *Skeletron* user study is always below the 50th percentile for all the task types, with the exception of *Card Sorting* that has value of 27.88 for the 75th percentile. Therefore, the workload experienced by the participants during the user study seems to be acceptable.

When analyzing each of the 6 items separately we note that *Mental Demand* is consistently the one that participants scored higher. This is understandable since the focus of this user study was mainly on having participants learn and understand the *YanuX Framework* through our initial briefing and the various exercises which require attention and mental effort. In fact, *Effort* had the second highest score which is consistent with that assessment. On the other hand, *Physical Demand* has the lowest score since our session required little to no physical activity from our participants, i.e., they just had to seat in front of a computer and use the provided smartphone.

The *Frustration* score is low possibly because participants were accompanied and guided towards finding the solution to the exercises. Similarly, the *Performance* score was low since everyone was eventually able to complete the exercises (note that lower scores mean *Perfect Performance* and high scores mean *Failure* in completing tasks). Finally, *Temporal Demand* has a relatively high score and a large variability. This is probably due to the fact that each of our sessions took between 45 min to 1 h, which is a relatively long time. Moreover, participants probably had a very wide range opinions about how they felt about being exposed to a lot of information during that time.

7 Conclusions and Future Work

This chapter presents the conclusions and final remarks about the research work conducted as part of this dissertation, including all related publications that were produced. It also presents ideas for future work.

7.1 Conclusions

The research work conducted as part of this dissertation aimed to explore the possibilities opened by the presence of several computing devices around us in our daily lives. The main objective was to find new and better ways for applications to take advantage of their presence instead of being confined to run on a single device at a time. This led us to the definition of the four Research Questions presented in section 1.4.

In order to address the first research question (**RQ$_1$** – "How to create and provide tools that enable developers to build applications that take advantage of co-located devices?"), we set out to find a way to provide those tools to developers. This led to the creation of the *YanuX Framework* which provides models and tools that allow developers to create applications that have the UI distributed across co-located devices. The framework provides a way for UI components to be automatically distributed across the available devices but the distribution can be easily modified by users according to their preferences. The framework also allows users to manage and share multiple application states in order to collaborate with other users present in the same environment.

The aforementioned capabilities stem from the desire to respond to the second research question (**RQ$_2$** – "How should the application state and graphical user interface be seamlessly distributed across co-located devices to support collaborative environments?"). In the case of the application state, our application adopted the approach of encapsulating it into a serializable JSON object that we called a *resource*. These *resources* are synchronized across the application instances running on multiple devices. This allows for the user interface to be kept updated and coherent across the multiple devices that are displaying user interface components. Collaboration between multiple users could also be achieved by sharing the *resources*, which encapsulate the application state with other users along with the devices of the users that have access to the *resource*.

This approach worked well across our application prototypes presented in Chapter 5. It was also well received and understood by the participants of our developer focused user study presented in section 6.4 since the vast majority of the participants had a strong level of agreement towards following this approach (see $DS1$ on Table 6.10 and Figure 6.13). Therefore, this approach responds to $\mathbf{RQ_2}$.

The aforementioned developer-focused study aimed to evaluate most of the technical aspects of the *YanuX Framework* and, as can be read in the results of the study (see 6.4.3.2), the feedback provided by the participants was largely positive. Not only from the point of view of the interest in the concept of better integrating co-located devices by providing an user experience in which the interface of the applications is spread among those devices, but also when it comes to the tools that our framework provided to implement them. Therefore, the *YanuX Framework* constitutes a response to $\mathbf{RQ_1}$ which, according to the results of our developer-focused study, should be able to cover the essential needs of developers that may be interested in pursuing the development of this novel type of applications. Nevertheless, based on some suggestions made by the participants, and our observations during the study, there are a few improvements that may be incorporated in future revisions of the framework to improve its learnability and usability.

Regarding the automatic distribution of UI components, we decided to distribute them according to the capabilities of the devices and their position in the environment in relation to each other. The capabilities can be automatically extracted using an application created for that purpose (see section 3.2.6). The detection of the position of the devices in the environment is related to our third research question ($\mathbf{RQ_3}$) – "How to capture the proxemic relationships between users and devices to associate devices in a cross-device interaction environment?"). We had to develop our own solution to respond to that question since there is no available solution that can be directly used to provide proxemic information, such as the estimates of the relative distances between devices and their orientation.

The resulting *Indoor Positioning System for Pervasive Environments* presented in Chapter 4 was created as a standalone solution that was successfully integrated with the *YanuX Framework* to provide the information required to establish proxemic relationships between devices. We can consider that the underlying IPS responded positively to the users' expectations since, in general, users agreed that the applications reacted to the presence of other devices during the user studies quickly enough Nevertheless, our IPS has some limitations given that we intended to use technologies that were already commonly available as the basis for its operation. For instance, the *proximity* turned out to be the most useful technique for applications based on the *YanuX Framework*. In fact, during the user studies, we used the *proximity* technique to get estimates of the distances between devices instead of inferring those distances from absolute positions by using one of the other techniques.

The IPS also supports absolute positioning through *fingerprinting* and *trilateration*.

However, these techniques underperformed when used in conjunction with the *YanuX Framework*. That may be partially attributed to the fact that absolute positions had to be used to calculate the relative positions required by the framework in order to establish proxemic relationships. Therefore, the mostly acceptable errors that we observed during our offline performance tests (see section 4.8) were potentially amplified when calculating the euclidean distance between two position estimates, since each one already had a significant error associated to it.

Part of the issues with the IPS accuracy may stem from the fact that the employed technologies (i.e., *Wi-Fi* and BLE) were not primarily created with indoor positioning in mind. Therefore, the information that is possible to obtain about the position of the devices will always be relatively inaccurate and a best effort approximation based on noisy or incomplete data. This allowed the solution to be easily deployed without additional hardware infrastructure, other than the usage of relatively inexpensive BLE beacons. Nevertheless, if the accepted trade-offs and compromises were different the proposed solution could have been a different one.

The *YanuX Framework* and the *Indoor Positioning System for Pervasive Environments* presented in Chapters 3 and 4, respectively, are the two major contributions that can be reused and improved upon by others in the future. However, the application prototypes that we developed along with the framework and that were presented in Chapter 5, along with the corresponding user studies based on them in Chapter 6, are also important contributions that may give insights to researchers that decide to follow on this research area in the future. Moreover, they played a central role when it came to responding to $\mathbf{RQ_4}$, i.e., “Are applications running across co-located devices beneficial to end-users under certain circumstances?”, by allowing us to present our vision to potential users in order to gather feedback and insights, thus allowing us to validate our research work.

The application prototypes were largely well received by participants of our user studies for the three end-user focused prototypes, i.e., the YanuX YouTube Viewer, YanuX Calculator and JuxtBoard. In general, the participants were very enthusiastic about the possibilities that are opened up by this new type of applications. The quantitative results of the user studies were also positive with good SUS scores, highly rated domain specific questions and other metrics were also mostly positive. Therefore, the response to $\mathbf{RQ_4}$ seems to be affirmative. In fact, the JuxtBoard user study (see 6.3) targeted that question by putting a single device scenario against a cross-device scenario while using the same application. The results presented in Table 6.6 and Figure 6.7 show us that in certain situations users preferred to use the cross-device mode, e.g., when they need to show content to other users and when they need to collaborate with them. This means that, at least under certain circumstances, the usage of cross-device applications should be beneficial for end-users.

Finally, by responding to the four aforementioned research questions we also ended up realizing the research statement that was presented in section 1.4, i.e., “Propose models and tools to assist developers with the creation of applications which have their UI distributed across multiple co-located devices based on their capabilities and proxemic relationships". Despite the core part of this realization being the *YanuX Framework*, since it contains the models and the tools mentioned in the research statement, it would not have been possible to achieve it without the contribution of the *Indoor Positioning System for Pervasive Environments* to capture the proxemic relationships of the devices. Moreover, the application prototypes allowed us to continuously test our progress and to conduct user studies with potential users.

7.2 Publications

Publishing is a very important part of scientific research. It allows us to present, discuss and collect feedback about our research. Publication of articles on peer-reviewed conference proceedings and academic journals also helps us to attest the quality and relevance of our contributions within the scientific community. Therefore, since the beginning of the thesis that we have been publishing our ideas and results as we progress.

The following publications presents results of the current research work:

1. **"Enabling the development of pervasive multi-device applications"** (Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '17) – This paper gives an overview of ideas and some of the initial progress made in the scope of this dissertation [San17]. It at the *EICS 2017 Doctoral Consortium* and the feedback that we got was that it was an ambitious proposition but that the foundation of our ideas is solid and that we should pursue with our work.

2. **"Designing a Framework to Support the Development of Smart Cross-device Applications"** (Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia - MUM 2018) – This paper presented the initial version of our framework detailing its architecture and components [SMC18]. It also introduced a application demo that was configured to distribute UI components across multiple devices using a simpler and less expressive approach than the one that is currently part of our framework. The paper was presented at *MUM 2018* and the feedback that we got was mostly positive.

3. **"YanuX - Pervasive distribution of the user interface by co-located devices"** (Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services) – This paper was presented at *MobiQuitous 2019* and further refined the framework and introduced the automatic distribution of UI components across multiple co-located devices by matching their capabilities with the requirements of the components [SMC19]. The proximity awareness of the devices was determined by the strength of BLE signals as described in 3.2.4. We then presented an application to playback *YouTube* videos that

takes advantage of these new features by distributing the components that display, control and select the video (see 5.1).

4. **"Computational Framework to Support Development of Applications Running on Multiple Co-located Devices"** (Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '21) – This paper consists of a technical note published at *EICS 2021* that introduces the framework in detail [San+21a]. The core of the framework itself remained the same, but it was extended to support saving multiple application states (*resources*) per users and per application. Moreover, these *resources* can be shared with other users to enable collaboration. Two *web components* were introduced to allow developers to easily provide resource management and component distribution to their users. The paper summarizes the API specification of all major components so that developers can grasp how the framework works. The indoor positioning system, which has been redesigned and expanded upon, is also briefly explained.

5. **"Applications across Co-located Devices: User Interface Distribution, Application State Management and Collaboration"** (Proceedings of the 19th International Conference on Advances in Mobile Computing & Multimedia) – This paper has been accepted for publishing at *MoMM 2021* [SMC21a]. It focuses on giving an overview of the current architecture, components and capabilities of the *YanuX Framework* with a focus on how users can redistribute the UI components, manage application state and collaborate in cross-device environments. The JuxtBoard prototype is presented as an example application that incorporate these features. This prototype was used as part of a user study to assess how participants react to these type of applications.

6. **"Designing Proxemic-aware Cross-Device Applications : A Feasibility Study"** (20th International Conference on Mobile and Ubiquitous Multimedia (MUM 2021)) – This paper has been accepted at *MUM 2021* [SMC21b]. It presents two of our prototypes (YanuX Calculator and JuxtBoard) and the user studies that were conducted with potential users to assess the feasibility, usability and usefulness of cross-device applications.

7. **"Indoor Positioning System for Ubiquitous Computing Environments"** (Intelligent Data Engineering and Automated Learning – IDEAL 2021) – This paper has been accepted at *IDEAL 2021* [San+21b]. It presents the indoor positioning system that was integrated into the *YanuX Framework* in order to determine when devices can be considered to be co-located so that the cross-device applications can span their UI across them. This system was previous presented in Chapter 4.

The developer-focused study presented in section 6.4 has yet to be part of a publication. Therefore, we still intend to use that study and finalize some work in the near future that should serve as a basis for more publications in a journal or conference.

7.3 Future Work

The foreseen contributions for this dissertation presented in section 1.5) have been fully realized and the research work addressed the four research questions presented in section 1.4. Moreover, given that the application prototypes were well received and the results of our user studies were mostly positive, we find that it is feasible to keep further developing the concept of applications that spread their user interface across multiple co-located devices. To do so, we will keep using and improving the framework and the indoor positioning system. Some of the improvements that we wish to make are based on user feedback gathered during the user studies performed with application prototypes. Others are more focused on details that need to be fine tuned in the core components of the framework based on the feedback and observations that we gathered during the developer-focused study.

For instance, we intend to refine the framework to support more richer interaction scenarios by enabling the distribution of UI components and integration between distinct applications on multiple devices, instead of focusing on one application at a time like we do know. The YanuX Coordinator could be extended to listen to events coming from the YanuX Broker and to react accordingly by managing applications, splitting the screen for simultaneous multitasking in public displays, or dealing with cross-device interaction gestures.

It should also be interesting to study how to improve the collaboration capabilities of the framework and how to make them more robust. Moreover, it is important to better address privacy issues since we are dealing with environments where multiple users and non-users may see what is displayed in public and semi-public screens. Therefore, the framework may be extended to take into consideration how privacy sensitive are certain pieces of information displayed on UI components, or the whole UI components themselves, when it comes to deciding where to show information. For instance, it may only show privacy sensitive information on personal devices of the user that owns that information. Moreover, instead of continuously enabling these privacy protection measures, we may also extend our IPS solution to use computer vision to monitor an interactive space to only enable those measures when needed.

We plan to develop more complex applications and continue studying how people can benefit from applications running across multiple co-located devices. We intend that some of those applications respond to societal challenges by taking advantage of the framework's capabilities. We are already applying the concept in the development of a solution towards Industry 5.0, which complements the Industry 4.0 paradigm in a transition to a sustainable, human-centric and resilient industry. Industry 5.0 will not reduce human value, but rather increase it through human–machine collaboration

[DY21], having in personalization a key factor. Therefore, it is essential to create smart HCI solutions that enhance, potentiate, the aforementioned collaboration.

We will continue to conduct more studies with developers as they develop real applications to evaluate if they can take advantage of the capabilities of the framework to build compelling applications with a rich user experience. For instance, there is previous research into UI prototyping of applications with cross-device compatibility and distribution in mind that can be possibly adapted to be used with our framework [LL08; Neb+14a]. There is also research on the testability of these kind of applications, which poses specific challenges that are not addressed by regular testing tools [Hus+16]. In the future, we may include these types of tools as part of our framework.

We plan to keep improving the indoor positioning system and to take better advantage of its absolute positioning capabilities. For instance, a radar widget could be used to represent the multiple devices in the environment and the components associated with them. Redistribution of components could be accomplished by dragging and dropping them around combined with other gestures.

We believe that we should still be able to improve the accuracy and response times of the IPS. It is still possible to explore better ways to process data during the *online* and *offline phases*, and to improve the usage of the available machine learning algorithms through better parameterization or by customizing them to better fit this application domain. Furthermore, there is currently no direct relationship between the position of a device at moment t and at $t+1$ because the computing of each position is stateless, which means that subsequent position estimates may diverge from the previous one by an unrealistic amount (e.g., distance between positions is larger than it would be possible at walking speed) which may lead to a poor user experience under some circumstances. This effect may be attenuated if we are able to incorporate information about previous estimates into the current one, e.g., by using Long Short-Term Memory (LSTM) neural networks or Hidden Markov Models (HMM). Moreover, there are also some new emerging technologies that are becoming more easily available and have the potential of reaching higher levels of accuracy (e.g., UWB and *Wi-Fi Round Trip Time (RTT)*.

Bibliography

[aba21] @abandonware/noble Contributors. *@abandonware/noble*. 2021-11. URL: https://www.npmjs.com/package/@abandonware/noble (visited on 2021-11-07) (cit. on p. 102).

[AI11] S. A. Ahson and M. Ilya. *Location-Based Services Handbook: Applications, Technologies, and Security*. Ed. by S. A. Ahson and M. Ilya. CRC Press, 2011. ISBN: 9781420071986 (cit. on pp. 45, 47, 60).

[Ake88] D. Akerberg. "Properties of a TDMA pico cellular office communication system". In: *IEEE 39th Vehicular Technology Conference*. IEEE, 1988, pp. 186–191. DOI: 10.1109/VETEC.1989.40071. URL: http://ieeexplore.ieee.org/document/40071/ (cit. on p. 57).

[AN07] S. Ali and P. Nobles. "A Novel Indoor Location Sensing Mechanism for IEEE 802.11 b/g Wireless LAN". In: *2007 4th Workshop on Positioning, Navigation and Communication* 2007 (2007-03), pp. 9–15. DOI: 10.1109/WPNC.2007.353605. URL: http://ieeexplore.ieee.org/document/4167811/ (cit. on p. 57).

[Alt+11] F. Alt et al. "Designing Shared Public Display Networks – Implications from Today's Paper-Based Notice Areas". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6696 LNCS. Springer Berlin Heidelberg, 2011, pp. 258–275. ISBN: 9783642217258. DOI: 10.1007/978-3-642-21726-5_17. URL: http://link.springer.com/10.1007/978-3-642-21726-5_17 (cit. on p. 3).

[Ame] S. American. *What is 'fuzzy logic'? Are there computers that are inherently fuzzy and do not apply the usual binary logic? - Scientific American*. URL: https://www.scientificamerican.com/article/what-is-fuzzy-logic-are-t/ (cit. on p. 90).

[App18] Apple Inc. *iBeacon*. 2018. URL: https://developer.apple.com/ibeacon/ (visited on 2018-08-27) (cit. on pp. 51, 58, 102).

[App] Apple Inc. *Handoff for Developers*. URL: https://developer.apple.com/handoff/ (visited on 2019-05-22) (cit. on p. 27).

[BP00] P. Bahl and V. Padmanabhan. "RADAR: an in-building RF-based user location and tracking system". In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 2. IEEE, 2000, pp. 775–784. ISBN: 0-7803-5880-5. DOI: 10.1109/INFCOM.2000.832252. URL: http://ieeexplore.ieee.org/document/832252/ (cit. on pp. 47, 54, 56, 57).

[Bal+03] H. Balakrishnan et al. "Lessons from developing and deploying the cricket indoor location system". 2003 (cit. on p. 62).

[Bal+05] H. Balakrishnan et al. *The Cricket Indoor Location System - User Manual*. Tech. rep. 2005. Cambridge, Massachusetts: MIT Computer Science and Artificial Intelligence Lab, 2005, p. 57. URL: http://cricket.csail.mit.edu/v2man.pdf (cit. on p. 62).

[BMG10] T. Ballendat, N. Marquardt, and S. Greenberg. "Proxemic interaction: designing for a proximity and orientation-aware environment". In: *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*. New York, New York, USA: ACM Press, 2010-11, p. 121. ISBN: 9781450303996. DOI: 10.1145/1936652.1936676. URL: http://dl.acm.org/citation.cfm?id=1936652.1936676 (cit. on pp. 38, 40, 42).

[BKM09] A. Bangor, P. Kortum, and J. Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale". In: *Journal of Usability Studies* 4 (2009), pp. 114–123. URL: http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS_Bangor_May2009.pdf (cit. on pp. 135, 140, 144, 168).

[BPP17] M. Barsotti, F. Paternò, and F. Pulina. "A web framework for cross-device gestures between personal devices and public displays". In: *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*. New York, NY, USA: ACM, 2017-11, pp. 69–78. ISBN: 9781450353786. DOI: 10.1145/3152832.3152858. URL: https://dl.acm.org/doi/10.1145/3152832.3152858 (cit. on pp. 19, 21, 26).

[BK16] M. Berkman and D. Karahoca. "Re-assessing the usability metric for user experience (UMUX) scale". In: *Journal of Usability Studies* 11.3 (2016), pp. 89–109. ISSN: 1931-3357. URL: https://uxpajournal.org/assessing-usability-metric-umux-scale/ (cit. on p. 168).

[Bla+17] G. de Blasio et al. "Study on an indoor positioning system for harsh environments based on Wi-Fi and bluetooth low energy". In: *Sensors (Switzerland)* 17.6 (2017-06), p. 1299. ISSN: 14248220. DOI: 10.3390/s17061299. URL: http://www.mdpi.com/1424-8220/17/6/1299 (cit. on p. 56).

[Blu15] Bluetooth SIG. *Proximity Profile 1.0.1*. Tech. rep. Bluetooth SIG, 2015, p. 20. URL: https://www.bluetooth.com/specifications/specs/proximity-profile-1-0-1/ (cit. on p. 51).

[Blu19] Bluetooth SIG. *Enhancing Bluetooth Location Services with Direction Finding*. 2019. URL: https://www.bluetooth.com/bluetooth-resources/enhancing-bluetooth-location-services-with-direction-finding/ (cit. on pp. 44, 51).

[Bor+14] S. Boring et al. "The Dark Patterns of Proxemic Sensing". In: *Computer* 47.8 (2014-08), pp. 56–60. ISSN: 0018-9162. DOI: 10.1109/MC.2014.223. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6879743 (cit. on p. 42).

[Bor+15] S. Borsci et al. "Assessing User Satisfaction in the Era of User Experience: Comparison of the SUS, UMUX, and UMUX-LITE as a Function of Product Experience". In: *International Journal of Human-Computer Interaction* 31.8 (2015-08), pp. 484–495. ISSN: 15327590. DOI: 10.1080/10447318.2015.1064648. URL: https://www.tandfonline.com/doi/full/10.1080/10447318.2015.1064648 (cit. on p. 168).

[BA15] J. Bradley and N. Agarwal. *Proof Key for Code Exchange by OAuth Public Clients*. Tech. rep. 7636. 2015-09, pp. 1–20. DOI: 10.17487/RFC7636. URL: https://www.rfc-editor.org/info/rfc7636 (cit. on pp. 66, 86).

[Bre+17] R. F. Brena et al. "Evolution of Indoor Positioning Technologies: A Survey". In: *Journal of Sensors* 2017 (2017-03), pp. 1–21. ISSN: 1687-725X. DOI: 10.1155/2017/2630413. URL: https://www.hindawi.com/journals/js/2017/2630413/ (cit. on pp. 43, 44, 48, 50, 51, 53, 60, 61).

[Bro96] J. Brooke. "SUS - A quick and dirty usability scale". In: *Usability Evaluation in Industry*. Ed. by P. W. Jordan et al. 1st. CRC Press, 1996. Chap. 21, pp. 189–194. ISBN: 9780748404605 (cit. on pp. 134, 138, 140, 144, 168).

[Bru+14] F. Brudy et al. "Is Anyone Looking? Mitigating Shoulder Surfing on Public Displays through Awareness and Protection". In: *Proceedings of The International Symposium on Pervasive Displays - PerDis '14*. New York, New York, USA: ACM Press, 2014-06, pp. 1–6. ISBN: 9781450329521. DOI: 10.1145/2611009.2611028. URL: http://dl.acm.org/citation.cfm?id=2611009.2611028 (cit. on p. 42).

[Bru01] P. Brusilovsky. "Adaptive Hypermedia". In: *User Modeling and User-Adapted Interaction* 11.1-2 (2001), pp. 87–110. ISSN: 09241868. DOI: 10.1023/A:1011143116306. URL: https://link.springer.com/article/10.1023/A:1011143116306 (cit. on p. 34).

[Buy+02] O. Buyukkokten et al. "Efficient web browsing on handheld devices using page and form summarization". In: *ACM Transactions on Information Systems* 20.1 (2002-01), pp. 82–115. ISSN: 1046-8188. DOI: 10.1145/503104.503109. URL: https://dl.acm.org/doi/10.1145/503104.503109 (cit. on p. 34).

[Byr+95] R. H. Byrd et al. "A Limited Memory Algorithm for Bound Constrained Optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995-09), pp. 1190–1208. ISSN: 1064-8275. DOI: 10.1137/0916069. URL: http://epubs.siam.org/doi/10.1137/0916069 (cit. on p. 118).

[Caf02] J. J. Caffery. *Wireless Location in CDMA Cellular Radio Systems*. Vol. 535. The Kluwer International Series in Engineering and Computer Science. Boston: Kluwer Academic Publishers, 2002, p. 189. ISBN: 0-7923-7703-6. DOI: 10.1007/b117784. URL: http://link.springer.com/10.1007/b117784 (cit. on p. 57).

[Cam21] Cambridge University Press. *MEASUREMENT | meaning in the Cambridge English Dictionary*. 2021. URL: https://dictionary.cambridge.org/dictionary/english/measurement (visited on 2020-10-08) (cit. on p. 44).

[CR14] B. Cardoso and T. Romão. "Presenting EveWorks, a framework for daily life event detection". In: *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems - EICS '14*. New York, New York, USA: ACM Press, 2014, pp. 289–294. ISBN: 9781450327251. DOI: 10.1145/2607023.2610279. URL: http://dl.acm.org/citation.cfm?doid=2607023.2610279 (cit. on p. 34).

[CR15] B. Cardoso and T. Romão. "Making Sense of EveXL, a DSL for Context Awareness". In: *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2015-08. ISBN: 978-1-63190-072-3. DOI: 10.4108/eai.22-7-2015.2260303. URL: http://eudl.eu/doi/10.4108/eai.22-7-2015.2260303 (cit. on p. 34).

[Cen+15] P. Centieiro et al. "In sync with fair play!: delivering a synchronized and cheat-preventing second screen gaming experience". In: *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology - ACE '15*. New York, New York, USA: ACM Press, 2015, pp. 1–11. ISBN: 9781450338523. DOI: 10.1145/2832932.2832953. URL: http://dl.acm.org/citation.cfm?doid=2832932.2832953 (cit. on p. 3).

[CY05] X. Chai and Q. Yang. "Reducing the Calibration Effort for Location Estimation Using Unlabeled Samples". In: *Third IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2005, pp. 95–104. ISBN: 0-7695-2299-8. DOI: 10.1109/PERCOM.2005.34. URL: http://ieeexplore.ieee.org/document/1392746/ (cit. on p. 54).

[CL11] T.-H. Chang and Y. Li. “Deep shot: a framework for migrating tasks across devices using mobile phone cameras”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI ’11*. New York, New York, USA: ACM Press, 2011-05, p. 2163. ISBN: 9781450302289. DOI: 10.1145/1978942.1979257. URL: http://dl.acm.org/citation.cfm?id=1978942.1979257 (cit. on pp. 16, 17, 26).

[Che+05] Y.-C. Chen et al. “Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics”. In: *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems - MSWiM ’05*. New York, New York, USA: ACM Press, 2005, p. 118. ISBN: 1595931880. DOI: 10.1145/1089444.1089466. URL: http://dl.acm.org/citation.cfm?id=1089466 (cit. on p. 55).

[CL17] X. Chen and Y. Li. “Improv: An input framework for improvising cross-device interaction by demonstration”. In: *ACM Transactions on Computer-Human Interaction* 24.2 (2017-05), pp. 1–21. ISSN: 15577325. DOI: 10.1145/3057862. URL: https://dl.acm.org/doi/10.1145/3057862 (cit. on p. 21).

[Che+14] X. A. Chen et al. “Duet: Exploring joint interactions on a smart phone and a smart watch”. In: *Conference on Human Factors in Computing Systems - Proceedings*. New York, NY, USA: ACM, 2014-04, pp. 159–168. ISBN: 9781450324731. DOI: 10.1145/2556288.2556955. URL: https://dl.acm.org/doi/10.1145/2556288.2556955 (cit. on p. 33).

[CL15] P.-Y. Y. Chi and Y. Li. “Weave: Scripting cross-device wearable interaction”. In: *Conference on Human Factors in Computing Systems - Proceedings*. Vol. 2015-April. New York, NY, USA: ACM, 2015-04, pp. 3923–3932. ISBN: 9781450331456. DOI: 10.1145/2702123.2702451. URL: https://dl.acm.org/doi/10.1145/2702123.2702451 (cit. on p. 33).

[Cli13] S. Clinch. “Smartphones and Pervasive Public Displays”. In: *IEEE Pervasive Computing* 12.1 (2013), pp. 92–95. ISSN: 1536-1268. DOI: 10.1109/MPRV.2013.16. URL: http://ieeexplore.ieee.org/document/6415936/ (cit. on p. 3).

[Con+03] K. Coninx et al. “Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Ed. by L. Chittaro. Vol. 2795. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 256–270. ISBN: 9783540452331. DOI: 10.1007/978-3-540-45233-1_19. URL: http://link.springer.com/10.1007/978-3-540-45233-1_19 (cit. on p. 27).

[Cou+95] J. Coutaz et al. “Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The Care Properties”. In: Springer, Boston, MA, 1995, pp. 115–120. doi: 10.1007/978-1-5041-2896-4_19. url: https://link.springer.com/chapter/10.1007/978-1-5041-2896-4_19 (cit. on p. 28).

[Cru+17] M. Cruz et al. “Exploring the use of second screen devices during live sports broadcasts to promote social interaction”. In: *Proceedings of the 14th Conference on Advances in Computer Entertainment Technology - ACE ’17*. Vol. 10714 LNCS. Springer, Cham, 2017-12, pp. 318–338. isbn: 9783319762692. doi: 10.1007/978-3-319-76270-8_23. url: http://link.springer.com/10.1007/978-3-319-76270-8_23 (cit. on p. 3).

[D H12] E. D. Hardt. *The OAuth 2.0 Authorization Framework*. Tech. rep. Microsoft, 2012-10. doi: 10.17487/rfc6749. url: https://www.rfc-editor.org/info/rfc6749 (cit. on pp. 66, 86).

[Dav+09] N. Davies et al. “Using bluetooth device names to support interaction in smart environments”. In: *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*. New York, New York, USA: ACM Press, 2009, p. 151. isbn: 978-1-60558-566-6. doi: 10.1145/1555816.1555832. url: http://portal.acm.org/citation.cfm?doid=1555816.1555832 (cit. on p. 3).

[DP08] D. Dearman and J. S. Pierce. “It’s on my other computer!: computing with multiple devices”. In: *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI ’08*. New York, New York, USA: ACM Press, 2008-04, p. 767. isbn: 9781605580111. doi: 10.1145/1357054.1357177. url: http://dl.acm.org/citation.cfm?id=1357054.1357177 (cit. on p. 1).

[Dec21] Decawave. *Decawave*. 2021-09. url: https://www.decawave.com/ (visited on 2021-09-10) (cit. on p. 63).

[Dey01] A. K. Dey. “Understanding and Using Context”. In: *Personal and Ubiquitous Computing* 5.1 (2001-02), pp. 4–7. issn: 1617-4909. doi: 10.1007/s007790170019. url: http://link.springer.com/10.1007/s007790170019 (cit. on p. 34).

[DY21] M. Di Nardo and H. Yu. “Special Issue “Industry 5.0: The Prelude to the Sixth Industrial Revolution””. In: *Applied System Innovation* 4.3 (2021). issn: 2571-5577. doi: 10.3390/asi4030045. url: https://www.mdpi.com/2571-5577/4/3/45 (cit. on p. 179).

[DG10] R. Diaz-Marino and S. Greenberg. “The proximity toolkit and ViconFace: the video”. In: *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA ’10*. New York, New York, USA: ACM Press, 2010, p. 4793. isbn: 9781605589305. doi:

10.1145/1753846.1754233. URL: https://www.youtube.com/watch?v=BhT0QgTSddM%20http://portal.acm.org/citation.cfm?doid=1753846.1754233 (cit. on p. 39).

[Dos+14] J. Dostal et al. "SpiderEyes: designing attention- and proximity-aware collaborative interfaces for wall-sized displays". In: *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*. New York, New York, USA: ACM Press, 2014-02, pp. 143–152. ISBN: 9781450321846. DOI: 10.1145/2557500.2557541. URL: http://dl.acm.org/citation.cfm?id=2557500.2557541 (cit. on p. 41).

[DB07] S. Dowling and A. Barney. *iPhone Premieres This Friday Night at Apple Retail Stores*. 2007. URL: https://www.apple.com/newsroom/2007/06/28iPhone-Premieres-This-Friday-Night-at-Apple-Retail-Stores/ (visited on 2019-07-10) (cit. on p. 2).

[EMa11] EMarketer. *What Do TV-Social Media Multitaskers Talk About?* 2011. URL: http://www.emarketer.com/Article/What-Do-TV-Social-Media-Multitaskers-Talk-About/1008301 (visited on 2016-09-30) (cit. on p. 3).

[Est18] Estimote Inc. *Estimote Products*. 2018. URL: https://estimote.com/products/#products (visited on 2018-08-27) (cit. on p. 102).

[Est21] Estimote Inc. *Estimote*. 2021-09. URL: https://estimote.com/ (visited on 2021-09-10) (cit. on p. 63).

[Eur18] Eurostat. *Individuals - devices used to access the internet (ISOC_CI_DEV_I)*. 2018. URL: https://ec.europa.eu/eurostat/databrowser/view/ISOC_CI_DEV_I/default/table?lang=en (visited on 2021-02-09) (cit. on pp. 1, 2).

[EMN05] F. Evennou, F. Marx, and E. Novakov. "Map-aided indoor mobile positioning system using particle filter". In: *IEEE Wireless Communications and Networking Conference, 2005*. Vol. 4. IEEE, 2005, pp. 2490–2494. ISBN: 0-7803-8966-2. DOI: 10.1109/WCNC.2005.1424905. URL: http://ieeexplore.ieee.org/document/1424905/ (cit. on p. 54).

[FH14] R. Faragher and R. Harle. "An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications". In: *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)* (2014), pp. 201–210. ISSN: 2331-5911. URL: http://www.ion.org/publications/abstract.cfm?jp=p%7B%5C&%7DarticleID=12411 (cit. on pp. 50, 55, 84).

[FH15] R. Faragher and R. Harle. "Location Fingerprinting With Bluetooth Low Energy Beacons". In: *IEEE Journal on Selected Areas in Communications* 33.11 (2015-11), pp. 2418–2428. ISSN: 0733-8716. DOI: 10.1109/JSAC.2015.2430281. URL: http://ieeexplore.ieee.org/document/7103024/ (cit. on pp. 50, 51, 55, 56, 83, 92, 102, 103).

[Fea18] Feathers Contributors. *Feathers - Instant Realtime and REST APIs with Node.js*. 2018. URL: https://feathersjs.com/ (visited on 2018-08-27) (cit. on p. 68).

[Fel+03] S. Feldmann et al. "An Indoor Bluetooth-Based Positioning System: Concept, Implementation and Experimental Evaluation". In: *Proceedings of the International Conference on Wireless Networks, {ICWN} '03, June 23 - 26, 2003, Las Vegas, Nevada, {USA}*. Ed. by W. Zhuang et al. CSREA Press, 2003, pp. 109–113 (cit. on p. 58).

[FSK05] J. Figueiras, H.-P. Schwefel, and I. Kovacs. "Accuracy and timing aspects of location information based on signal-strength measurements in Bluetooth". In: *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*. Vol. 4. IEEE, 2005, pp. 2685–2690. ISBN: 978-3-8007-29. DOI: 10.1109/PIMRC.2005.1651931. URL: http://ieeexplore.ieee.org/document/1651931/ (cit. on p. 58).

[Fon+10] J. M. C. Fonseca et al. *Model-Based UI XG Final Report*. Tech. rep. W3C, 2010, p. 32. URL: https://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/ (cit. on p. 35).

[FP14] L. Frosini and F. Paternò. "User interface distribution in multi-device and multi-user environments with dynamically migrating engines". In: *The 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS 2014* (2014), pp. 55–64. DOI: 10.1145/2607023.2607032. URL: http://dl.acm.org/citation.cfm?id=2607032 (cit. on pp. 17, 20, 22, 32).

[GWW10] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. "Automatically generating personalized user interfaces with Supple". In: *Artificial Intelligence* 174.12-13 (2010-08), pp. 910–950. ISSN: 00043702. DOI: 10.1016/j.artint.2010.05.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0004370210000822 (cit. on p. 35).

[GP16] A. Gallidabino and C. Pautasso. "The Liquid.js Framework for Migrating and Cloning Stateful Web Components across Multiple Devices". In: *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*. New York, New York, USA: ACM Press, 2016, pp. 183–186. ISBN: 9781450341448. DOI: 10.1145/2872518.2890538. URL: http://dl.acm.org/citation.cfm?doid=2872518.2890538 (cit. on p. 27).

[Gal+16] A. Gallidabino et al. "On the Architecture of Liquid Software: Technology Alternatives and Design Space". In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016-04, pp. 122–127. ISBN: 978-1-5090-2131-4. DOI: 10.1109/WICSA.2016.14. URL: http://ieeexplore.ieee.org/document/7516819/ (cit. on pp. 1, 27, 77).

[Gar] Gartner. *The Internet of Things.* URL: https://www.gartner.com/en/information-technology/glossary/internet-of-things (visited on 2016-10-17) (cit. on p. 3).

[Gel+08] H. Gellersen et al. "Supporting device discovery and spontaneous interaction with spatial references". In: *Personal and Ubiquitous Computing* 13.4 (2008-07), pp. 255–264. ISSN: 1617-4909. DOI: 10.1007/s00779-008-0206-3. URL: http://dl.acm.org/citation.cfm?id=1527347.1527372 (cit. on pp. 28–30, 62).

[Gér] A. Géron. *Hands-on Machine Learning with Scikit-Learn , Keras & TensorFlow.* O'Reilly Media, Inc. ISBN: 9781492032649 (cit. on p. 111).

[GPS13] G. Ghiani, F. Paternò, and C. Santoro. "Interactive customization of ubiquitous Web applications". In: *Journal of Visual Languages & Computing* 24.1 (2013-02), pp. 37–52. ISSN: 1045926X. DOI: 10.1016/j.jvlc.2012.10.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S1045926X12000729 (cit. on p. 26).

[GP10] G. Ghiani and F. Paternò. "Supporting Mobile Users in Selecting Target Devices". In: *Journal of Universal Computer Science* 16.15 (2010-07), pp. 2019–2037. DOI: 10.3217/jucs-016-15-2019 (cit. on pp. 21, 31).

[GPS12] G. Ghiani, F. Paternò, and C. Santoro. "Push and pull of web user interfaces in multi-device environments". In: *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '12.* New York, New York, USA: ACM Press, 2012-05, p. 10. ISBN: 9781450312875. DOI: 10.1145/2254556.2254563. URL: http://dl.acm.org/citation.cfm?id=2254556.2254563 (cit. on pp. 16, 18, 20, 26).

[GPA13] G. Ghiani, J. Polet, and V. Antila. "Towards intelligent migration of user interfaces". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 8093 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 203–217. ISBN: 9783642402753. DOI: 10.1007/978-3-642-40276-0_16. URL: http://link.springer.com/10.1007/978-3-642-40276-0_16 (cit. on p. 26).

[Ghi+14] G. Ghiani et al. "Beyond responsive design: Context-dependent multimodal augmentation of web applications". In: *MobiWIS 2014: Mobile Web Information Systems.* Vol. 8640 LNCS. Springer, Cham, 2014, pp. 71–85. ISBN: 9783319103587. DOI: 10.1007/978-3-319-10359-4_6. URL: http://link.springer.com/10.1007/978-3-319-10359-4_6 (cit. on p. 19).

[Ghi+17] G. Ghiani et al. "Personalization of Context-Dependent Applications Through Trigger-Action Rules". In: *ACM Transactions on Computer-Human Interaction* 24.2 (2017-05), pp. 1–33. ISSN: 1073-0516. DOI: 10.1145/3057861. URL: https://dl.acm.org/doi/10.1145/3057861 (cit. on p. 35).

[GS14] F. Giglietto and D. Selva. “Second Screen and Participation: A Content Analysis on a Full Season Dataset of Tweets”. In: *Journal of Communication* 64.2 (2014-04), pp. 260–277. ISSN: 0021-9916. DOI: 10.1111/jcom.12085. URL: https://academic.oup.com/joc/article/64/2/260-277/4085976 (cit. on p. 3).

[Git18] GitHub Inc. *Electron - Build cross platform desktop apps with JavaScript, HTML, and CSS*. 2018. URL: https://electronjs.org/ (visited on 2018-08-28) (cit. on p. 68).

[GW16] L. Goode and T. Warren. *This is what Microsoft HoloLens is really like*. 2016. URL: https://www.theverge.com/2016/4/1/11334488/microsoft-hololens-video-augmented-reality-ar-headset-hands-on (visited on 2016-10-04) (cit. on p. 2).

[Goo12] Google. *The New Multi-screen World: Understanding Cross-platform Consumer Behavior*. 2012. URL: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/the-new-multi-screen-world-study/ (visited on 2016-10-03) (cit. on p. 3).

[Goo21a] Google Developers. *Motion sensors (Use the rotation vector sensor) - Android Developers*. 2021-09. URL: https://developer.android.com/guide/topics/sensors/sensors_motion#sensors-motion-rotate (visited on 2021-09-20) (cit. on pp. 83, 86).

[Goo21b] Google Developers. *SensorManager.getOrientation - Android Developers*. 2021-08. URL: https://developer.android.com/reference/android/hardware/SensorManager#getOrientation(float[],%20float[]) (visited on 2021-09-20) (cit. on pp. 83, 86).

[Goo18a] Google LLC. *Specification for Eddystone, an open beacon format from Google*. 2018. URL: https://github.com/google/eddystone (visited on 2019-07-22) (cit. on p. 58).

[Goo18b] Google LLC. *WebView*. 2018. URL: https://developer.android.com/reference/android/webkit/WebView (visited on 2018-08-28) (cit. on p. 68).

[Goo18c] Google LLC. *YouTube Player API Reference for iframe Embeds - YouTube IFrame Player API*. 2018. URL: https://developers.google.com/youtube/iframe_api_reference (visited on 2019-06-19) (cit. on p. 121).

[GK99] S. Greenberg and H. Kuzuoka. “Using digital but physical surrogates to mediate awareness, communication and privacy in media spaces”. In: *Personal Technologies* 3.4 (1999-12), pp. 182–198. ISSN: 0949-2054. DOI: 10.1007/BF01540552. URL: http://link.springer.com/10.1007/BF01540552 (cit. on p. 39).

[Gre+11] S. Greenberg et al. "Proxemic interactions: the new ubicomp?" In: *interactions* 18.1 (2011-01), p. 42. ISSN: 10725520. DOI: 10.1145/1897239.1897250. URL: http://portal.acm.org/citation.cfm?doid=1897239.1897250 (cit. on pp. 3, 35, 38–40, 42, 75).

[Gre+14] S. Greenberg et al. "Dark patterns in proxemic interactions". In: *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*. New York, New York, USA: ACM Press, 2014-06, pp. 523–532. ISBN: 9781450329026. DOI: 10.1145/2598510.2598541. URL: http://dl.acm.org/citation.cfm?id=2598510.2598541 (cit. on p. 42).

[Gri15] R. A. Grier. "How High is High? A Meta-Analysis of NASA-TLX Global Workload Scores". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 59.1 (2015-09), pp. 1727–1731. ISSN: 2169-5067. DOI: 10.1177/1541931215591373. URL: https://journals.sagepub.com/doi/10.1177/1541931215591373 (cit. on pp. 170, 171).

[GSM18] GSMArena.com. *Xiaomi Mi 8 - Full phone specifications*. 2018. URL: https://www.gsmarena.com/xiaomi_mi_8-9065.php (visited on 2019-07-26) (cit. on p. 79).

[GLN09] Y. Gu, A. Lo, and I. Niemegeers. "A survey of indoor positioning systems for wireless personal networks". In: *IEEE Communications Surveys & Tutorials* 11.1 (2009), pp. 13–32. ISSN: 1553-877X. DOI: 10.1109/SURV.2009.090103. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4796924 (cit. on p. 45).

[Hae+04] A. Haeberlen et al. "Practical robust localization over large-scale 802.11 wireless networks". In: *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*. New York, New York, USA: ACM Press, 2004, p. 70. ISBN: 1581138687. DOI: 10.1145/1023720.1023728. URL: http://portal.acm.org/citation.cfm?doid=1023720.1023728 (cit. on p. 47).

[Hal90] E. T. Hall. *The Hidden Dimension*. 27th ed. Anchor Books, 1990, p. 217. ISBN: 9780844665528. URL: https://books.google.com/books?id=HlmqAAAACAAJ%7B%5C&%7Dpgis=1 (cit. on pp. 36, 37, 77).

[HW14] P. Hamilton and D. J. Wigdor. "Conductor: Enabling and Understanding Cross-Device Interaction". In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. New York, New York, USA: ACM Press, 2014-04, pp. 2773–2782. ISBN: 9781450324731. DOI: 10.1145/2556288.2557170. URL: http://dl.acm.org/citation.cfm?id=2556288.2557170 (cit. on p. 32).

[Ham+05] A. Hampapur et al. "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking". In: *IEEE Signal Processing Magazine* 22.2 (2005-03), pp. 38–51. ISSN: 1053-5888. DOI: 10.1109/MSP.2005.1406476. URL: http://ieeexplore.ieee.org/document/1406476/ (cit. on p. 48).

[HN10] N. Harrison and K. Natalie. *iPad Arrives This Saturday*. 2010. URL: https://www.apple.com/newsroom/2010/03/29iPad-Arrives-This-Saturday/ (visited on 2019-07-10) (cit. on p. 2).

[Har06] S. G. Hart. "NASA-task load index (NASA-TLX); 20 years later". In: *Proceedings of the Human Factors and Ergonomics Society*. Vol. 50. 9. SAGE PublicationsSage CA: Los Angeles, CA, 2006-10, pp. 904–908. ISBN: 9780945289296. DOI: 10.1177/154193120605000909. URL: https://journals.sagepub.com/doi/10.1177/154193120605000909 (cit. on pp. 152, 170).

[HS88] S. G. Hart and L. E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Advances in Psychology* 52.C (1988-01), pp. 139–183. ISSN: 01664115. DOI: 10.1016/S0166-4115(08)62386-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166411508623869 (cit. on pp. 152, 170).

[Har+99a] A. Harter et al. "The anatomy of a context-aware application". In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking - MobiCom '99*. MobiCom '99. New York, New York, USA: ACM Press, 1999, pp. 59–68. ISBN: 1581131429. DOI: 10.1145/313451.313476. URL: http://doi.acm.org/10.1145/313451.313476 (cit. on p. 62).

[HMP96] J. Hartman, U. Manber, and L. Peterson. *Liquid software: A new paradigm for networked systems*. Tech. rep. Tucson, AZ, USA, 1996 (cit. on pp. 26, 27).

[Har+99b] J. J. Hartman et al. "Joust: A Platform for Liquid Software". In: *Computer* 32.4 (1999-04), pp. 50–56. ISSN: 00189162. DOI: 10.1109/2.755005. URL: http://ieeexplore.ieee.org/document/755005/ (cit. on p. 27).

[HSK04] M. Hazas, J. Scott, and J. Krumm. "Location-aware computing comes of age". In: *Computer* 37.2 (2004-02), pp. 95–97. ISSN: 0018-9162. DOI: 10.1109/MC.2004.1266301. URL: http://ieeexplore.ieee.org/document/1266301/ (cit. on p. 58).

[Hey13] R. Heydon. *Bluetooth Low Energy: The Developer's Handbook*. Upper Saddle River, N.J: Prentice Hall, 2013. ISBN: 978-0132888363 (cit. on p. 50).

[HBW00] J. Hightower, G. Borriello, and R. Want. "SpotON: An indoor 3D location sensing technology based on RF signal strength". In: *UW CSE 2000* August (2000), p. 16. URL: http://www.hightowerweb.org/pubs/hightower2000indoor/hightower2000indoor.pdf (cit. on p. 58).

[Hor+19] T. Horak et al. “Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI ’19*. New York, New York, USA: ACM Press, 2019, pp. 1–13. ISBN: 9781450359702. DOI: 10.1145/3290605.3300846. URL: http://dl.acm.org/citation.cfm?doid=3290605.3300846 (cit. on p. 28).

[Hos+07] A. M. Hossain et al. “Indoor Localization Using Multiple Wireless Technologies”. In: *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*. IEEE, 2007-10, pp. 1–8. ISBN: 978-1-4244-1454-3. DOI: 10.1109/MOBHOC.2007.4428622. URL: http://ieeexplore.ieee.org/document/4428622/ (cit. on p. 47).

[HM15] S. Houben and N. Marquardt. “WatchConnect: A toolkit for prototyping smartwatch-centric cross-device applications”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Vol. 2015-April. New York, NY, USA: ACM, 2015-04, pp. 1247–1256. ISBN: 9781450331456. DOI: 10.1145/2702123.2702215. URL: https://dl.acm.org/doi/10.1145/2702123.2702215 (cit. on p. 33).

[Hu+04] W. Hu et al. “A Survey on Visual Surveillance of Object Motion and Behaviors”. In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 34.3 (2004-08), pp. 334–352. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2004.829274. URL: http://ieeexplore.ieee.org/document/1310448/ (cit. on p. 48).

[Hus+14] M. Husmann et al. “MultiMasher: Providing Architectural Support and Visual Tools for Multi-device Mashups”. In: *Web Information Systems Engineering – WISE 2014*. Vol. 8787. Springer, Cham, 2014-10, pp. 199–214. DOI: 10.1007/978-3-319-11746-1_15. URL: https://link.springer.com/chapter/10.1007/978-3-319-11746-1_15 (cit. on p. 36).

[Hus+16] M. Husmann et al. “UI Testing Cross-Device Applications”. In: *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ISS ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 179–188. ISBN: 9781450342483. DOI: 10.1145/2992154.2992177. URL: https://doi.org/10.1145/2992154.2992177 (cit. on p. 179).

[Int15] International Telecommunication Union. *Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 100 GHz*. Tech. rep. International Telecommunication Union, 2015. URL: https://www.itu.int/rec/R-REC-P.1238-8-201507-I/en (cit. on p. 57).

[Jak+13] M. R. Jakobsen et al. “Information Visualization and Proxemics: Design Opportunities and Empirical Findings”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013-12), pp. 2386–2395. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.166. URL: http://ieeexplore.ieee.org/document/6634094/ (cit. on p. 43).

[Jia+14] Z. Jianyong et al. “RSSI based Bluetooth low energy indoor positioning”. In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014-10, pp. 526–533. ISBN: 978-1-4673-8054-6. DOI: 10.1109/IPIN.2014.7275525. URL: http://ieeexplore.ieee.org/document/7275525/ (cit. on p. 51).

[Jok+15] T. Jokela et al. “A Comparison of Methods to Move Visual Objects Between Personal Mobile Devices in Different Contexts of Use”. In: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. New York, NY, USA: ACM, 2015-08, pp. 172–181. ISBN: 9781450336529. DOI: 10.1145/2785830.2785841. URL: https://dl.acm.org/doi/10.1145/2785830.2785841 (cit. on pp. 21, 32).

[Jos+08] R. José et al. “Instant Places: Using Bluetooth for Situated Interaction in Public Displays”. In: *IEEE Pervasive Computing* 7.4 (2008-10), pp. 52–57. ISSN: 15361268. DOI: 10.1109/MPRV.2008.74. URL: http://ieeexplore.ieee.org/document/4653472/ (cit. on p. 3).

[JLK08] W. Ju, B. A. Lee, and S. R. Klemmer. “Range: Exploring Implicit Interaction through Electronic Whiteboard Design”. In: *Proceedings of the ACM 2008 conference on Computer supported cooperative work - CSCW ’08*. New York, New York, USA: ACM Press, 2008-11, p. 17. ISBN: 9781605580074. DOI: 10.1145/1460563.1460569. URL: http://dl.acm.org/citation.cfm?id=1460563.1460569 (cit. on p. 41).

[JLK07] W. G. Ju, B. A. Lee, and S. R. Klemmer. “Range: Exploring Proxemics in Collaborative Whiteboard Interaction Wendy”. In: *CHI ’07 extended abstracts on Human factors in computing systems - CHI ’07*. New York, New York, USA: ACM Press, 2007-04, p. 2483. ISBN: 9781595936424. DOI: 10.1145/1240866.1241028. URL: http://dl.acm.org/citation.cfm?id=1240866.1241028 (cit. on p. 41).

[KK04] K. Kaemarungsi and P. Krishnamurthy. “Modeling of indoor positioning systems based on location fingerprinting”. In: *IEEE INFOCOM 2004*. Vol. 2. C. IEEE, 2004, pp. 1012–1022. ISBN: 0-7803-8355-9. DOI: 10.1109/INFCOM.2004.1356988. URL: http://ieeexplore.ieee.org/document/1356988/ (cit. on p. 55).

[KP16] A. A. Kalbandhe and S. C. Patil. "Indoor Positioning System using Bluetooth Low Energy". In: *2016 International Conference on Computing, Analytics and Security Trends (CAST)*. IEEE, 2016-12, pp. 451–455. ISBN: 978-1-5090-1338-8. DOI: 10.1109/CAST.2016.7915011. URL: http://ieeexplore.ieee.org/document/7915011/ (cit. on p. 59).

[Kam21] Kamalshadi. *kamalshadi/Localization: Multilateration and triangulation for target localization*. 2021-11. URL: https://github.com/kamalshadi/Localization (visited on 2021-11-08) (cit. on p. 118).

[Kan19] Kantar TNS Germany. *The Connected Consumer*. 2019. URL: https://www.google.com/publicdata/explore?ds=dg8d1eetcqsb1_ (visited on 2019-05-21) (cit. on p. 1).

[KM] J. M. Keenan and A. J. Motley. "Radio coverage in buildings". eng. In: *British Telecom technology journal* 8.1 (), pp. 19–24. ISSN: 0265-0193. URL: http://cat.inist.fr/?aModele=afficheN%7B%5C&%7Dcpsidt=6928371 (cit. on p. 57).

[Kim+15] D.-Y. Kim et al. "Accurate Indoor Proximity Zone Detection Based on Time Window and Frequency with Bluetooth Low Energy". In: *Procedia Computer Science* 56.1 (2015), pp. 88–95. ISSN: 18770509. DOI: 10.1016/j.procs.2015.07.199. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050915016804 (cit. on pp. 59, 84).

[Kin+06] T. King et al. "COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses". In: *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization - WiNTECH '06*. September. New York, New York, USA: ACM Press, 2006, p. 34. ISBN: 1595935400. DOI: 10.1145/1160987.1160995. URL: http://portal.acm.org/citation.cfm?doid=1160987.1160995 (cit. on p. 54).

[Klo+15] C. N. Klokmose et al. "Webstrates: Shareable Dynamic Media". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. New York, New York, USA: ACM Press, 2015, pp. 280–290. ISBN: 9781450337793. DOI: 10.1145/2807442.2807446. URL: http://dl.acm.org/citation.cfm?doid=2807442.2807446 (cit. on p. 28).

[KKB14] C. N. Klokmose, M. Korn, and H. Blunck. "WiFi proximity detection in mobile web applications". In: *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems - EICS '14* (2014), pp. 123–128. DOI: 10.1145/2607023.2610281. URL: http://dl.acm.org/citation.cfm?doid=2607023.2610281 (cit. on p. 59).

[KO13] V. Kostakos and T. Ojala. “Public Displays Invade Urban Spaces”. In: *IEEE Pervasive Computing* 12.April (2013), pp. 8–13. ISSN: 1536-1268. DOI: 10.1109/MPRV.2013.15. URL: https://ieeexplore.ieee.org/document/6415929 (cit. on p. 3).

[KTH00] KTH Royal Institute of Technology. *WIPS Technical Documentation*. Tech. rep. Stockholm, Sweden: KTH Royal Institute of Technology, 2000. URL: https://archive.ssvl.kth.se/csd/0012/technical.pdf (cit. on p. 60).

[KK11] C. E. Kulkarni and S. R. Klemmer. “Automatically adapting web pages to heterogeneous devices”. In: *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA ’11*. New York, New York, USA: ACM Press, 2011-05, p. 1573. ISBN: 9781450302685. DOI: 10.1145/1979742.1979810. URL: http://portal.acm.org/citation.cfm?doid=1979742.1979810 (cit. on p. 35).

[Küp05] A. Küpper. *Location-Based Services*. Chichester, UK: John Wiley & Sons, Ltd, 2005-08, pp. 1–365. ISBN: 9780470092330. DOI: 10.1002/0470092335. URL: http://doi.wiley.com/10.1002/0470092335 (cit. on pp. 43, 45, 46, 60, 62).

[KPV11] A. Kushki, K. Plataniotis, and A. Venetsanopoulos. *WLAN Positioning Systems*. 2000. Cambridge: Cambridge University Press, 2011, p. 160. ISBN: 9780511978784. DOI: 10.1017/CBO9780511978784. URL: http://ebooks.cambridge.org/ref/id/CBO9780511978784 (cit. on pp. 46–48, 60).

[LHS08] B. Laugwitz, T. Held, and M. Schrepp. “Construction and Evaluation of a User Experience Questionnaire”. In: *HCI and Usability for Education and Work*. Ed. by A. Holzinger. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 63–76. ISBN: 978-3-540-89350-9 (cit. on p. 138).

[LB15] N. Leahy and A. Bessete. *Apple Watch In-Store Preview & Online Pre-Order Begin Friday*. 2015. URL: https://www.apple.com/newsroom/2015/04/09Apple-Watch-In-Store-Preview-Online-Pre-Order-Begin-Friday/ (visited on 2019-07-10) (cit. on p. 2).

[LS18] J. R. Lewis and J. Sauro. “Item Benchmarks for the System Usability Scale”. In: *Journal of Usability Studies* 13.3 (2018), pp. 158–167. URL: http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS_Lewis_May2018.pdf (cit. on pp. 135, 168, 169).

[LUM13] J. R. Lewis, B. S. Utesch, and D. E. Maher. “UMUX-LITE - When there’s no time for the SUS”. In: *Conference on Human Factors in Computing Systems - Proceedings*. New York, NY, USA: ACM, 2013-04, pp. 2099–2102. ISBN: 9781450318990. DOI: 10.1145/2470654.2481287. URL: https://dl.acm.org/doi/10.1145/2470654.2481287 (cit. on pp. 152, 154, 168).

[LUM15] J. R. Lewis, B. S. Utesch, and D. E. Maher. "Investigating the correspondence between UMUX-LITE and SUS scores". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9186. Springer, Cham, 2015, pp. 204–211. ISBN: 9783319208855. DOI: 10.1007/978-3-319-20886-2_20. URL: https://link.springer.com/chapter/10.1007/978-3-319-20886-2_20%20http://link.springer.com/10.1007/978-3-319-20886-2_20 (cit. on pp. 152, 154, 168).

[Lim+06] H. Lim et al. "Zero-Configuration, Robust Indoor Localization: Theory and Experimentation". In: *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. IEEE, 2006, pp. 1–12. ISBN: 1-4244-0221-2. DOI: 10.1109/INFOCOM.2006.223. URL: http://ieeexplore.ieee.org/document/4146876/ (cit. on p. 57).

[LL08] J. Lin and J. A. Landay. "Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1313–1322. ISBN: 9781605580111. DOI: 10.1145/1357054.1357260. URL: https://doi.org/10.1145/1357054.1357260 (cit. on p. 179).

[Lio+04] M. N. Lionel et al. "LANDMARC: Indoor location sensing using active RFID". In: *Wireless Networks*. Vol. 10. 6. Kluwer Academic Publishers, 2004-11, pp. 701–710. ISBN: 0-7695-1893-1. DOI: 10.1023/B:WINE.0000044029.06344.dd. URL: http://link.springer.com/10.1023/B:WINE.0000044029.06344.dd (cit. on p. 58).

[Löc11] M. Löchtefeld. "Advanced interaction with mobile projection interfaces". In: *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology - UIST '11 Adjunct*. New York, New York, USA: ACM Press, 2011-10, p. 43. ISBN: 9781450310147. DOI: 10.1145/2046396.2046415. URL: http://dl.acm.org/citation.cfm?id=2046396.2046415 (cit. on p. 41).

[Lou21] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. iii).

[MA75] E. H. Mamdani and S. Assilian. "An experiment in linguistic synthesis with a fuzzy logic controller". In: *International Journal of Man-Machine Studies* 7.1 (1975-01), pp. 1–13. ISSN: 00207373. DOI: 10.1016/S0020-7373(75)80002-2. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020737375800022 (cit. on p. 91).

[MP16] M. Manca and F. Paternò. "Customizable dynamic user interface distribution". In: *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems.* New York, NY, USA: ACM, 2016-06, pp. 27–37. ISBN: 9781450343220. DOI: 10.1145/2933242.2933259. URL: https://dl.acm.org/doi/10.1145/2933242.2933259 (cit. on pp. 17, 28).

[MK11] E. Marcotte and J. Keith. *Responsive web design.* A Book Apart, 2011, p. 153. ISBN: 9781937557188. URL: https://abookapart.com/products/responsive-web-design (cit. on pp. 27, 35).

[Mar11] N. Marquardt. "Proxemic interactions in ubiquitous computing ecologies". In: *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11.* New York, New York, USA: ACM Press, 2011-05, p. 1033. ISBN: 9781450302685. DOI: 10.1145/1979742.1979691. URL: http://dl.acm.org/citation.cfm?id=1979742.1979691 (cit. on pp. 9, 39).

[MG12] N. Marquardt and S. Greenberg. "Informing the Design of Proxemic Interactions". In: *IEEE Pervasive Computing* 11.2 (2012-02), pp. 14–23. ISSN: 1536-1268. DOI: 10.1109/MPRV.2012.15. URL: http://ieeexplore.ieee.org/document/6127852/ (cit. on pp. 40, 43).

[MHG12] N. Marquardt, K. Hinckley, and S. Greenberg. "Cross-device interaction via micro-mobility and f-formations". In: *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12.* New York, New York, USA: ACM Press, 2012-10, p. 13. ISBN: 9781450315807. DOI: 10.1145/2380116.2380121. URL: http://dl.acm.org/citation.cfm?id=2380116.2380121 (cit. on pp. 31, 38, 39).

[Mar+11] N. Marquardt et al. "The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11.* New York, New York, USA: ACM Press, 2011-10, p. 315. ISBN: 9781450307161. DOI: 10.1145/2047196.2047238. URL: http://dl.acm.org/citation.cfm?id=2047196.2047238 (cit. on pp. 9, 19, 22, 35, 39).

[Mar+12] N. Marquardt et al. "Gradual engagement: facilitating information exchange between digital devices as a function of proximity". In: *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces - ITS '12.* New York, New York, USA: ACM Press, 2012-11, p. 31. ISBN: 9781450312097. DOI: 10.1145/2396636.2396642. URL: http://dl.acm.org/citation.cfm?id=2396636.2396642 (cit. on p. 30).

[Mar+20] P. Martins et al. "Improving bluetooth beacon-based indoor location and fingerprinting". In: *Journal of Ambient Intelligence and Humanized Computing* 11.10 (2020-10), pp. 3907–3919. ISSN: 1868-5137. DOI: 10.1007/s12652-01

9-01626-2. URL: http://link.springer.com/10.1007/s12652-019-01626-2 (cit. on pp. 50, 51, 56, 58, 92).

[Mau12] R. Mautz. "Indoor Positioning Technologies". PhD thesis. 2012, p. 127. ISBN: 9783908440314; DOI: 10.3929/ethz-a-007313554. URL: http://e-collection.library.ethz.ch/eserv/eth:5659/eth-5659-01.pdf (cit. on pp. 43, 48, 52, 53, 63).

[Maz+09] S. Mazuelas et al. "Robust Indoor Positioning Provided by Real-Time RSSI Values in Unmodified WLAN Networks". In: *IEEE Journal of Selected Topics in Signal Processing* 3.5 (2009-10), pp. 821–831. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2009.2029191. URL: http://ieeexplore.ieee.org/document/5290370/ (cit. on p. 57).

[McD05] P. McDermott-Wells. *What is Bluetooth?* 2005-01. DOI: 10.1109/MP.2005.1368913. URL: http://ieeexplore.ieee.org/document/1368913/ (cit. on p. 50).

[MVV11] J. Melchior, J. Vanderdonckt, and P. Van Roy. "A model-based approach for distributed user interfaces". In: *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '11.* New York, New York, USA: ACM Press, 2011-06, p. 11. ISBN: 9781450306706. DOI: 10.1145/1996461.1996488. URL: http://dl.acm.org/citation.cfm?id=1996461.1996488 (cit. on p. 19).

[Mel+09] J. Melchior et al. "A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications". In: *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS '09.* New York, New York, USA: ACM Press, 2009-07, p. 69. ISBN: 9781605586007. DOI: 10.1145/1570433.1570449. URL: http://dl.acm.org/citation.cfm?id=1570433.1570449 (cit. on pp. 20, 33).

[Mes+08] J. Meskens et al. "Gummy for multi-platform user interface designs". In: *Proceedings of the working conference on Advanced visual interfaces - AVI '08.* New York, New York, USA: ACM Press, 2008-05, p. 233. ISBN: 9781605581415. DOI: 10.1145/1385569.1385607. URL: http://portal.acm.org/citation.cfm?doid=1385569.1385607 (cit. on p. 35).

[Mic17] Microsoft Corporation. *TypeScript - JavaScript that scales.* 2017. URL: https://www.typescriptlang.org/ (visited on 2018-08-27) (cit. on p. 68).

[MSP15] T. Mikkonen, K. Systä, and C. Pautasso. "Towards Liquid Web Applications". In: *15th International Conference on Web Engineering ICWE 2015.* Vol. 9114. Rotterdam, the Netherlands: Springer, Cham, 2015, pp. 134–143. ISBN: 9783319198897. DOI: 10.1007/978-3-319-19890-3_10. URL: http://link.springer.com/10.1007/978-3-319-19890-3_10 (cit. on p. 27).

[MM15] A. Moreira and F. Meneses. "Where@UM - Dependable organic radio maps". In: *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2015-10, pp. 1–9. ISBN: 978-1-4673-8402-5. DOI: 10.1109/IPIN.2015.7346751. URL: http://ieeexplore.ieee.org/document/7346751/ (cit. on p. 54).

[Mue+14] F. Mueller et al. "Proxemics play: understanding proxemics for designing digital play experiences". In: *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*. New York, New York, USA: ACM Press, 2014-06, pp. 533–542. ISBN: 9781450329026. DOI: 10.1145/2598510.2598532. URL: http://dl.acm.org/citation.cfm?doid=2598510.2598532 (cit. on p. 40).

[MJ15] P. Mukherjee and B. J. Jansen. "Correlation of Brand Mentions in Social Media and Web Searching Before and After Real Life Events: Phase Analysis of Social Media and Search Data for Super Bowl 2015 Commercials". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015-11, pp. 21–26. ISBN: 978-1-4673-8493-3. DOI: 10.1109/ICDMW.2015.60. URL: http://ieeexplore.ieee.org/document/7395647/ (cit. on p. 3).

[Nac+05] M. A. Nacenta et al. "A comparison of techniques for multi-display reaching". In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*. New York, New York, USA: ACM Press, 2005-04, p. 371. ISBN: 1581139985. DOI: 10.1145/1054972.1055024. URL: http://dl.acm.org/citation.cfm?id=1054972.1055024 (cit. on p. 13).

[NAS21] NASA. *TLX @ NASA Ames - NASA TLX Paper/Pencil Version*. 2021-10. URL: https://humansystems.arc.nasa.gov/groups/TLX/tlxpaperpencil.php (visited on 2021-10-21) (cit. on pp. 152, 170).

[Neb17] M. Nebeling. "XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. New York, New York, USA: ACM Press, 2017, pp. 4574–4584. ISBN: 9781450346559. DOI: 10.1145/3025453.3025547. URL: http://dl.acm.org/citation.cfm?doid=3025453.3025547 (cit. on pp. 20, 28).

[ND16] M. Nebeling and A. K. Dey. "XDBrowser: User-Defined Cross-Device Web Page Designs". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. New York, New York, USA: ACM Press, 2016, pp. 5494–5505. ISBN: 9781450333627. DOI: 10.1145/2858036.2858048. URL: http://dl.acm.org/citation.cfm?doid=2858036.2858048 (cit. on p. 20).

[NSN13] M. Nebeling, M. Speicher, and M. C. Norrie. "CrowdAdapt: enabling crowdsourced web page adaptation for individual viewing conditions and preferences". In: *Proceedings of the 5th ACM SIGCHI symposium on Engineering* *interactive computing systems - EICS '13*. New York, New York, USA: ACM Press, 2013, p. 23. ISBN: 9781450321389. DOI: 10.1145/2494603.2480304. URL: http://dl.acm.org/citation.cfm?doid=2494603.2480304 (cit. on p. 34).

[Neb+14a] M. Nebeling et al. "Interactive Development of Cross-Device User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 2793–2802. ISBN: 9781450324731. DOI: 10.1145/2556288.2556980. URL: https://doi.org/10.1145/2556288.2556980 (cit. on p. 179).

[Neb+14b] M. Nebeling et al. "Interactive development of cross-device user interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2014-04, pp. 2793–2802. ISBN: 9781450324731. DOI: 10.1145/2556288.2556980. URL: https://dl.acm.org/doi/10.1145/2556288.2556980 (cit. on p. 36).

[Neb+14c] M. Nebeling et al. "XDKinect: Development Framework for Cross-Device Interaction using Kinect". In: *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems - EICS '14*. New York, New York, USA: ACM Press, 2014-06, pp. 65–74. ISBN: 9781450327251. DOI: 10.1145/2607023.2607024. URL: http://dl.acm.org/citation.cfm?doid=2607023.2607024 (cit. on p. 41).

[Neb+15] M. Nebeling et al. "XDSession: Integrated development and testing of cross-device applications". In: *EICS 2015 - Proceedings of the 2015 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. New York, NY, USA: ACM, 2015-06, pp. 22–27. ISBN: 9781450336468. DOI: 10.1145/2774225.2775075. URL: https://dl.acm.org/doi/10.1145/2774225.2775075 (cit. on p. 36).

[Nie11] Nielsen. *In the U.S., Tablets are TV Buddies while eReaders Make Great Bedfellows*. 2011. URL: http://www.nielsen.com/us/en/insights/news/2011/in-the-u-s-tablets-are-tv-buddies-while-ereaders-make-great-bedfellows.html (visited on 2016-09-30) (cit. on p. 3).

[OLL09] J.-O. Oh, M.-S. Lee, and S. Lee. "An Acoustic-Based Relative Positioning System for Multiple Mobile Devices". In: *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*. IEEE, 2009, pp. 1565–1570. ISBN: 978-1-4244-5244-6. DOI: 10.1109/ICCIT.2009.103. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5369929 (cit. on pp. 62, 63).

[Oru+18] F. Orujov et al. "Smartphone based intelligent indoor positioning using fuzzy logic". In: *Future Generation Computer Systems* 89 (2018-12), pp. 335–348. ISSN: 0167739X. DOI: 10.1016/j.future.2018.06.030. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X18305004 (cit. on pp. 59, 76, 90, 92).

[Pae+04] T. Paek et al. "Toward universal mobile interaction for shared displays". In: *Proceedings of the 2004 ACM conference on Computer supported cooperative work - CSCW '04*. New York, New York, USA: ACM Press, 2004-11, p. 266. ISBN: 1581138105. DOI: 10.1145/1031607.1031649. URL: http://dl.acm.org/citation.cfm?id=1031607.1031649 (cit. on p. 3).

[Par+18] S. Park et al. "AdaM: Adapting Multi-User Interfaces for Collaborative Environments in Real-Time". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. New York, New York, USA: ACM Press, 2018-03, pp. 1–14. ISBN: 9781450356206. DOI: 10.1145/3173574.3173758. arXiv: 1803.01166. URL: http://dl.acm.org/citation.cfm?doid=3173574.3173758 (cit. on pp. 18, 22, 28, 77).

[Pat20] F. Paternò. "Concepts and design space for a better understanding of multi-device user interfaces". In: *Universal Access in the Information Society* 19.2 (2020-06), pp. 409–432. ISSN: 16155297. DOI: 10.1007/s10209-019-00650-5. URL: http://link.springer.com/10.1007/s10209-019-00650-5 (cit. on pp. 14, 15, 20, 21, 27, 34, 35).

[PS12] F. Paternò and C. Santoro. "A logical framework for multi-device user interfaces". In: *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '12*. EICS '12. New York, New York, USA: ACM Press, 2012-06, p. 45. ISBN: 9781450311687. DOI: 10.1145/2305484.2305494. URL: http://dl.acm.org/citation.cfm?id=2305484.2305494%20https://doi.org/10.1145/2305484.2305494%20http://dl.acm.org/citation.cfm?doid=2305484.2305494 (cit. on p. 16).

[PSS11] F. Paternò, C. Santoro, and L. D. Spano. "Engineering the authoring of usable service front ends". In: *Journal of Systems and Software* 84.10 (2011-10), pp. 1806–1822. ISSN: 01641212. DOI: 10.1016/j.jss.2011.05.025. URL: https://linkinghub.elsevier.com/retrieve/pii/S0164121211001257 (cit. on p. 35).

[PSS09] F. Paterno', C. Santoro, and L. D. Spano. "MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments". In: *ACM Transactions on Computer-Human Interaction* 16.4 (2009-11), pp. 1–30. ISSN: 10730516. DOI: 10.1145/1614390.1614394. URL: http://dl.acm.org/citation.cfm?id=1614390.1614394 (cit. on p. 28).

[POY01] N. Patwari, R. O'Dea, and Yanwei Wang. "Relative location in wireless networks". In: *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202)*. Vol. 2. IEEE, 2001, pp. 1149–1153. ISBN: 0-7803-6728-6. DOI: 10.1109/VETECS.2001.944560. URL: http://ieeexplore.ieee.org/document/944560/ (cit. on p. 57).

[Pen+07] C. Peng et al. "BeepBeep: a high accuracy acoustic ranging system using COTS mobile devices". In: *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*. New York, New York, USA: ACM Press, 2007-11, p. 1. ISBN: 9781595937636. DOI: 10.1145/1322263.1322265. URL: http://dl.acm.org/citation.cfm?id=1322263.1322265 (cit. on p. 62).

[PVB04] P. Perez, J. Vermaak, and A. Blake. "Data Fusion for Visual Tracking With Particles". In: *Proceedings of the IEEE* 92.3 (2004-03), pp. 495–513. ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.823147. URL: http://ieeexplore.ieee.org/document/1271403/ (cit. on p. 48).

[Pet18] C. Pettey. *3 Reasons Why VR and AR Are Slow to Take Off - Smarter With Gartner*. 2018. URL: https://www.gartner.com/smarterwithgartner/3-reasons-why-vr-and-ar-are-slow-to-take-off/ (visited on 2019-07-10) (cit. on p. 2).

[Poz21] Pozyx NV. *Pozyx*. 2021-09. URL: https://pozyx.io/ (visited on 2021-09-10) (cit. on p. 63).

[Pri] Princeton. *Fuzzy Sets and Pattern Recognition*. URL: https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm (cit. on p. 91).

[PCB00] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. "The Cricket location-support system". In: *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*. Vol. 2000. MobiCom '00 August. New York, New York, USA: ACM Press, 2000, pp. 32–43. ISBN: 1581131976. DOI: 10.1145/345910.345917. URL: http://doi.acm.org/10.1145/345910.345917 (cit. on p. 62).

[Pri+01] N. B. Priyantha et al. "The cricket compass for context-aware mobile applications". In: *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01*. MobiCom '01 July. New York, New York, USA: ACM Press, 2001, pp. 1–14. ISBN: 1581134223. DOI: 10.1145/381677.381679. URL: http://doi.acm.org/10.1145/381677.381679 (cit. on p. 62).

[Rad] Radius Network. *Android Beacon Library*. URL: https://altbeacon.github.io/android-beacon-library/ (visited on 2021-03-29) (cit. on p. 102).

[Räd13] R. Rädle. "Design and evaluation of proxemics-aware environments to support navigation in large information spaces". In: *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*. New York, New York, USA: ACM Press, 2013-04, p. 1957. ISBN: 9781450319522. DOI: 10.1145/2468356.2468710. URL: http://dl.acm.org/citation.cfm?id=2468356.2468710 (cit. on p. 41).

[Räd+14] R. Rädle et al. "HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration". In: *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*. New York, New York, USA: ACM Press, 2014-11, pp. 45–54. ISBN: 9781450325875. DOI: 10.1145/2669485.2669500. URL: http://dl.acm.org/citation.cfm?id=2669485.2669500 (cit. on pp. 16, 32).

[Räd+15] R. Rädle et al. "Spatially-aware or spatially-agnostic? Elicitation and evaluation of user-defined cross-device interactions". In: *Conference on Human Factors in Computing Systems - Proceedings*. Vol. 2015-April. New York, NY, USA: ACM, 2015-04, pp. 3913–3922. ISBN: 9781450331456. DOI: 10.1145/2702123.2702287. URL: https://dl.acm.org/doi/10.1145/2702123.2702287 (cit. on p. 32).

[Rap01] T. S. Rappaport. *Wireless Communications: Principles and Practice*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001, p. 687. ISBN: 0130422320 (cit. on p. 57).

[Ras+12] M. Raspopoulos et al. "Cross device fingerprint-based positioning using 3D Ray Tracing". In: *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2012-08, pp. 147–152. ISBN: 978-1-4577-1379-8. DOI: 10.1109/IWCMC.2012.6314193. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6314193 (cit. on p. 47).

[Rei03] F. Reichheld, Frederick. "The one number you need to grow". In: *Harvard Business Review* December.4 (2003-12), pp. 46–54. ISSN: 00178012. URL: https://hbr.org/2003/12/the-one-number-you-need-to-grow (cit. on pp. 134, 152, 169).

[Ric15] J. Richer. "RFC 7662: OAuth 2.0 Token Introspection". In: *Rfc 7662* (2015) (cit. on p. 100).

[Rid+15] M. E. Rida et al. "Indoor Location Position Based on Bluetooth Signal Strength". In: *2015 2nd International Conference on Information Science and Control Engineering*. IEEE, 2015-04, pp. 769–773. ISBN: 978-1-4673-6850-6. DOI: 10.1109/ICISCE.2015.177. URL: http://ieeexplore.ieee.org/document/7120717/ (cit. on p. 58).

[Riv+12] F. Rivoal et al. *Media Queries*. 2012. URL: https://www.w3.org/TR/css3-mediaqueries/ (visited on 2021-08-23) (cit. on p. 36).

[Rod08] T. Rodden. “Living in a ubiquitous world”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1881 (2008-10), pp. 3837–3838. ISSN: 1364-503X. DOI: 10.1098/rsta.2008.0146. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2008.0146%20https://royalsocietypublishing.org/doi/10.1098/rsta.2008.0146 (cit. on p. 1).

[Ros+13] M. Rossi et al. “RoomSense”. In: *Proceedings of the 4th Augmented Human International Conference on - AH ’13*. New York, New York, USA: ACM Press, 2013, pp. 89–95. ISBN: 9781450319041. DOI: 10.1145/2459236.2459252. URL: http://dl.acm.org/citation.cfm?doid=2459236.2459252 (cit. on p. 63).

[REH04] N. Roussel, H. Evans, and H. Hansen. “MirrorSpace: Using Proximity as an Interface to Video-Mediated Communication”. In: *Proceedings of Second International Conference, PERVASIVE 2004*. Springer Berlin Heidelberg, 2004, pp. 345–350. DOI: 10.1007/978-3-540-24646-6_25. URL: http://link.springer.com/10.1007/978-3-540-24646-6_25 (cit. on p. 39).

[Rox+07] A. Roxin et al. “Survey of Wireless Geolocation Techniques”. In: *2007 IEEE Globecom Workshops*. IEEE, 2007-11, pp. 1–9. ISBN: 978-1-4244-2024-7. DOI: 10.1109/GLOCOMW.2007.4437809. URL: http://ieeexplore.ieee.org/document/4437809/ (cit. on p. 47).

[Rus+16] M. E. Rusli et al. “An Improved Indoor Positioning Algorithm Based on RSSI-Trilateration Technique for Internet of Things (IOT)”. In: *2016 International Conference on Computer and Communication Engineering (ICCCE)* (2016-07), pp. 72–77. DOI: 10.1109/ICCCE.2016.28. URL: http://ieeexplore.ieee.org/document/7808286/ (cit. on p. 58).

[Sah+03] S. Saha et al. “Location determination of a mobile device using IEEE 802.11b access point signals”. In: *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003*. Vol. 3. IEEE, 2003, pp. 1987–1992. ISBN: 0-7803-7700-1. DOI: 10.1109/WCNC.2003.1200692. URL: http://ieeexplore.ieee.org/document/1200692/ (cit. on p. 54).

[Sak+14] N. Sakimura et al. *OpenID Connect Core 1.0 incorporating errata set 1*. 2014. URL: http://openid.net/specs/openid-connect-core-1_0.html%20https://openid.net/specs/openid-connect-core-1_0.html (visited on 2018-08-27) (cit. on p. 66).

[San17] P. A. Santos. “Enabling the development of pervasive multi-device applications”. In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS ’17*. New York, New York, USA: ACM Press, 2017, pp. 153–156. ISBN: 9781450350839. DOI: 10.1145/3102113.3102156.

URL: http://dl.acm.org/citation.cfm?doid=3102113.3102156 (cit. on p. 176).

[SMC13] P. A. Santos, R. N. Madeira, and N. Correia. "FCT4U - When Private Mobile Displays Meet Public Situated Displays to Enhance the User Experience". In: *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. IEEE, 2013-12, pp. 186–193. ISBN: 978-1-4799-2482-0. DOI: 10.1109/UIC-ATC.2013.112. URL: https://ieeexplore.ieee.org/document/6726208 (cit. on pp. 3, 5).

[SMC18] P. A. Santos, R. N. Madeira, and N. Correia. "Designing a Framework to Support the Development of Smart Cross-device Applications". In: *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia - MUM 2018*. New York, New York, USA: ACM Press, 2018, pp. 367–374. ISBN: 9781450365949. DOI: 10.1145/3282894.3289727. URL: http://dl.acm.org/citation.cfm?doid=3282894.3289727 (cit. on p. 176).

[SMC19] P. A. Santos, R. N. Madeira, and N. Correia. "YanuX - Pervasive distribution of the user interface by co-located devices". In: *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. New York, NY, USA: ACM, 2019-11, pp. 368–377. ISBN: 9781450372831. DOI: 10.1145/3360774.3360832. URL: https://dl.acm.org/doi/10.1145/3360774.3360832 (cit. on p. 176).

[SMC21a] P. A. Santos, R. N. Madeira, and N. Correia. "Applications across Co-located Devices: User Interface Distribution, Application State Management and Collaboration". In: *Proceedings of the 19th International Conference on Advances in Mobile Computing & Multimedia*. Linz, Austria: Association for Computing Machinery, 2021. ISBN: 9781450395564. DOI: 10.1145/3487664.3487748 (cit. on p. 177).

[SMC21b] P. A. Santos, R. N. Madeira, and N. Correia. "Designing Proxemic-aware Cross-Device Applications : A Feasibility Study". In: *20th International Conference on Mobile and Ubiquitous Multimedia (MUM 2021)*. Leuven, Belgium: Association for Computing Machinery, 2021. ISBN: 9781450386432. DOI: 10.1145/3490632.3490658 (cit. on p. 177).

[San+21a] P. A. Santos et al. "Computational Framework to Support Development of Applications Running on Multiple Co-located Devices". In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '21*. New York, New York, USA: Association for Computing Machinery, 2021. ISBN: 9781450384490. DOI: 10.1145/3459926.3464758 (cit. on p. 177).

[San+21b] P. A. Santos et al. “Indoor Positioning System for Ubiquitous Computing Environments”. In: *Intelligent Data Engineering and Automated Learning – IDEAL 2021*. Ed. by H. Yin et al. Springer International Publishing, 2021. ISBN: 978-3-030-91607-7 (cit. on p. 177).

[SW13] S. Santosa and D. Wigdor. “A field study of multi-device workflows in distributed workspaces”. In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp ’13*. New York, New York, USA: ACM Press, 2013, p. 63. ISBN: 9781450317702. DOI: 10.1145/2493432.2493476. URL: http://dl.acm.org/citation.cfm?doid=2493432.2493476 (cit. on p. 1).

[Sch+13] G. Schiavo et al. “Sensing and reacting to users’ interest: an adaptive public display”. In: *CHI ’13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA ’13*. New York, New York, USA: ACM Press, 2013-04, p. 1545. ISBN: 9781450319522. DOI: 10.1145/2468356.2468632. URL: http://dl.acm.org/citation.cfm?id=2468356.2468632 (cit. on p. 41).

[Sch+15] M. Schreiner et al. “Connichiwa: A Framework for Cross-Device Web Applications”. In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA ’15*. New York, New York, USA: ACM Press, 2015-04, pp. 2163–2168. ISBN: 9781450331463. DOI: 10.1145/2702613.2732909. URL: http://dl.acm.org/citation.cfm?id=2702613.2732909 (cit. on pp. 20, 33).

[Sch19] M. Schrepp. *User Experience Questionnaire Handbook Version 8*. 2019. URL: https://www.ueq-online.org/Material/Handbook.pdf (cit. on p. 140).

[SHT14] M. Schrepp, A. Hinderks, and J. Thomaschewski. “Applying the User Experience Questionnaire (UEQ) in Different Evaluation Scenarios”. In: *Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience*. Ed. by A. Marcus. Cham: Springer International Publishing, 2014, pp. 383–392. ISBN: 978-3-319-07668-3 (cit. on p. 138).

[SR92] S. Y. Seidel and T. S. Rappaport. “914 MHz path loss prediction model for indoor wireless communication in multi floored buildings”. In: *IEEE Transactions on Antennas Propagation* 40.2 (1992), pp. 207–217 (cit. on p. 57).

[Soc18] Socket.IO Contributors. *Socket.IO*. 2018. URL: https://socket.io/ (visited on 2018-08-27) (cit. on p. 68).

[Soh+11] T. Sohn et al. “Myngle: Unifying and filtering web content for unplanned access between multiple personal devices”. In: *UbiComp’11 - Proceedings of the 2011 ACM Conference on Ubiquitous Computing*. New York, New York, USA: ACM Press, 2011, pp. 257–266. ISBN: 9781450309103. DOI: 10.1145/2030112.2030147. URL: http://dl.acm.org/citation.cfm?doid=2030112.2030147 (cit. on p. 25).

[SK13] H. Sørensen and J. Kjeldskov. "Moving Beyond Weak Identifiers for Proxemic Interaction". In: *Proceedings of International Conference on Advances in Mobile Computing & Multimedia - MoMM '13*. New York, New York, USA: ACM Press, 2013-12, pp. 18–22. ISBN: 9781450321068. DOI: 10.1145/2536853.2536910. URL: http://dl.acm.org/citation.cfm?id=2536853.2536910 (cit. on pp. 58, 64).

[Sør+13] H. Sørensen et al. "Proxemic interaction in a multi-room music system". In: *Proceedings of the 25th Australian Computer-Human Interaction Conference on Augmentation, Application, Innovation, Collaboration - OzCHI '13*. New York, New York, USA: ACM Press, 2013-11, pp. 153–162. ISBN: 9781450325257. DOI: 10.1145/2541016.2541046. URL: http://dl.acm.org/citation.cfm?id=2541016.2541046 (cit. on p. 40).

[Sør+14] H. Sørensen et al. "The 4C framework: Principles ofinteraction in digitalecosystems". In: *UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. New York, New York, USA: ACM Press, 2014, pp. 87–97. ISBN: 9781450329682. DOI: 10.1145/2632048.2636089. URL: https://doi.org/10.1145/2632048.2636089%20http://dl.acm.org/citation.cfm?doid=2632048.2636089 (cit. on pp. 22, 23).

[SVK14] R. Sukale, S. Voida, and O. Koval. "The proxemic web: designing for proxemic interactions with responsive web design". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*. New York, New York, USA: ACM Press, 2014-09, pp. 171–174. ISBN: 9781450330473. DOI: 10.1145/2638728.2638768. URL: http://dl.acm.org/citation.cfm?id=2638728.2638768 (cit. on p. 42).

[Tab+18] A. J. Tab et al. *CSS Flexible Box Layout Module Level 1*. 2018. URL: https://www.w3.org/TR/css-flexbox-1/ (visited on 2021-08-23) (cit. on p. 36).

[Tab+20] A. J. Tab et al. *CSS Grid Layout Module Level 1*. 2020. URL: https://www.w3.org/TR/css-grid-1/ (visited on 2021-08-23) (cit. on p. 36).

[TM15] A. Taivalsaari and T. Mikkonen. "From Apps to Liquid Multi-Device Software". In: *The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC-2015)*. 1. 2015, pp. 34–40. DOI: 10.1016/j.procs.2015.07.179. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050915016609 (cit. on p. 27).

[TMS14] A. Taivalsaari, T. Mikkonen, and K. Systa. "Liquid Software Manifesto: The Era of Multiple Device Ownership and Its Implications for Software Architecture". In: *2014 IEEE 38th Annual Computer Software and Applications Conference*. IEEE, 2014-07, pp. 338–343. ISBN: 978-1-4799-3575-8. DOI:

10.1109/COMPSAC.2014.56. URL: http://ieeexplore.ieee.org/document/6899235/ (cit. on pp. 13, 27).

[Tec20] D. Tech. *How Far Can You Go?* 2020. URL: http://www.davidgyoungtech.com/2020/05/15/how-far-can-you-go (visited on 2020-09-10) (cit. on p. 51).

[The21] The SciPy community. *scipy.optimize.minimize*. 2021-11. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html (visited on 2021-11-08) (cit. on p. 118).

[TC99] D. Thevenin and J. Coutaz. "Plasticity of User Interfaces: Framework and Research Agenda". In: *Human-Computer Interaction INTERACT '99: IFIP TC. 13*. Ed. by M. A. Sasse and C. Johnson. Amsterdam, The Netherlands: IOS Press, 1999, pp. 110–116. URL: http://iihm.imag.fr/publs/1999/interact99_plasticite.pdf (cit. on p. 77).

[Top13] J. Topolsky. *I used Google Glass: the future, but with monthly updates*. 2013. URL: https://www.theverge.com/2013/2/22/4013406/i-used-google-glass-its-the-future-with-monthly-updates (cit. on p. 2).

[Tut] Tutorialspoint. *Artificial Intelligence - Fuzzy Logic Systems - Tutorialspoint*. URL: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm (cit. on pp. 90, 91).

[Vai21] L. S. Vailshery. *Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025*. 2021-03. URL: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/ (visited on 2021-08-19) (cit. on p. 3).

[VC04] C. Vandervelpen and K. Coninx. "Towards model-based design support for distributed user interfaces". In: *Nordic Conference on Human-Computer Interaction; Vol. 82* (2004-10), p. 61. DOI: 10.1145/1028014.1028023. URL: http://dl.acm.org/citation.cfm?id=1028014.1028023 (cit. on p. 27).

[Var+07] A. Varshavsky et al. "GSM indoor localization". In: *Pervasive and Mobile Computing* 3.6 (2007-12), pp. 698–720. ISSN: 15741192. DOI: 10.1016/j.pmcj.2007.07.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S1574119207000478 (cit. on p. 47).

[Vir+20] P. Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (2020-03), pp. 261–272. ISSN: 1548-7091. DOI: 10.1038/s41592-019-0686-2. URL: http://www.nature.com/articles/s41592-019-0686-2 (cit. on pp. 118, 119).

[VB04] D. Vogel and R. Balakrishnan. "Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users". In: *Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04*. Vol. 6. 2. New York, New York, USA: ACM Press, 2004-10, p. 137. ISBN: 1581139578. DOI: 10.1145/1029632.1029656. URL: http://dl.acm.org/citation.cfm?id=1029656 (cit. on p. 41).

[WAM21] WAMP. *The Web Application Messaging Protocol — Web Application Messaging Protocol version 2 documentation*. 2021. URL: https://wamp-proto.org/ (visited on 2021-03-05) (cit. on pp. 76, 82).

[WHT03] L. Wang, W. Hu, and T. Tan. "Recent developments in human motion analysis". In: *Pattern Recognition* 36.3 (2003-03), pp. 585–601. ISSN: 00313203. DOI: 10.1016/S0031-3203(02)00100-0. URL: http://linkinghub.elsevier.com/retrieve/pii/S0031320302001000 (cit. on p. 48).

[WBG12] M. Wang, S. Boring, and S. Greenberg. "Proxemic peddler: a public advertising display that captures and preserves the attention of a passerby". In: *Proceedings of the 2012 International Symposium on Pervasive Displays - PerDis '12*. New York, New York, USA: ACM Press, 2012-06, pp. 1–6. ISBN: 9781450314145. DOI: 10.1145/2307798.2307801. URL: http://dl.acm.org/citation.cfm?id=2307798.2307801 (cit. on p. 40).

[Wan97] L.-X. Wang. *A Course in Fuzzy Systems and Control*. Prentice Hall, 1997, p. 448. ISBN: 9780135408827. URL: http://portal.acm.org/citation.cfm?id=248374%7B%5C&%7Ddl= (cit. on p. 91).

[Wan+03] Y. Wang et al. "An indoors wireless positioning system based on wireless local area network infrastructure". In: *6th Int. Symp. on Satellite Navigation Technology Including Mobile Positioning & Location Services*. 54. 2003 (cit. on p. 57).

[WS12] R. Want and B. N. Schilit. "Interactive Digital Signage". In: *Computer* 45.5 (2012-05), pp. 21–24. ISSN: 0018-9162. DOI: 10.1109/MC.2012.169. URL: http://ieeexplore.ieee.org/document/6197776/ (cit. on p. 3).

[Wan+92] R. Want et al. "The Active Badge Location System". In: *ACM Transactions on Information Systems* 10.1 (1992-01), pp. 91–102. ISSN: 1046-8188. DOI: 10.1145/128756.128759. URL: http://doi.acm.org/10.1145/128756.128759%20https://dl.acm.org/doi/10.1145/128756.128759 (cit. on p. 60).

[WJH97] A. Ward, A. Jones, and A. Hopper. "A new location technique for the active office". In: *IEEE Personal Communications* 4.5 (1997-12), pp. 42–47. ISSN: 10709916. DOI: 10.1109/98.626982. arXiv: arXiv:1011.1669v3. URL: http://ieeexplore.ieee.org/document/626982/ (cit. on p. 62).

[Wei91] M. Weiser. “The Computer for the 21st Century”. In: *Scientific American* 265 (1991), pp. 94–105. URL: https://www.jstor.org/stable/24938718 (cit. on p. 1).

[Wer14] M. Werner. *Indoor Location-Based Services*. Cham: Springer International Publishing, 2014, pp. 1–233. ISBN: 978-3-319-10698-4. DOI: 10.1007/978-3-319-10699-1. URL: http://link.springer.com/10.1007/978-3-319-10699-1 (cit. on pp. 46, 47, 62).

[WHA21] WHATWG. *HTML Living Standard - The slot element*. 2021-10. URL: https://html.spec.whatwg.org/multipage/scripting.html#the-slot-element (visited on 2021-10-17) (cit. on p. 131).

[Xia+17] S. Xia et al. “Indoor fingerprint positioning based on Wi-Fi: An overview”. In: *ISPRS International Journal of Geo-Information* 6.5 (2017-04), p. 135. ISSN: 22209964. DOI: 10.3390/ijgi6050135. URL: http://www.mdpi.com/2220-9964/6/5/135 (cit. on pp. 49, 50, 53, 55).

[YX09] B. Yan and L. Xiaochun. “Research on UWB indoor positioning based on TDOA technique”. In: *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE, 2009-08, pp. 1–167–1–170. ISBN: 978-1-4244-3863-1. DOI: 10.1109/ICEMI.2009.5274900. URL: http://ieeexplore.ieee.org/document/5274900/ (cit. on p. 63).

[YW14] J. Yang and D. Wigdor. “Panelrama: enabling easy specification of cross-device web applications”. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI ’14*. New York, New York, USA: ACM Press, 2014-04, pp. 2783–2792. ISBN: 9781450324731. DOI: 10.1145/2556288.2557199. URL: http://dl.acm.org/citation.cfm?id=2556288.2557199 (cit. on pp. 17, 18, 22, 28).

[YHV14] M. Yasir, S.-W. Ho, and B. N. Vellambi. “Indoor Positioning System Using Visible Light and Accelerometer”. In: *Journal of Lightwave Technology* 32.19 (2014-10), pp. 3306–3316. ISSN: 0733-8724. DOI: 10.1109/JLT.2014.2344772. URL: http://ieeexplore.ieee.org/document/6868970/ (cit. on p. 61).

[YG14] Z. Yorke and J. Greenwood. *Sports Fans and the Second Screen*. 2014. URL: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/sports-fans-and-the-second-screen/ (visited on 2016-10-03) (cit. on p. 3).

[YAU03] M. Youssef, A. Agrawala, and A. Udaya Shankar. “WLAN location determination via clustering and probability distributions”. In: *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003)*. PERCOM ’03. Washington, DC, USA: IEEE Comput. Soc, 2003, pp. 143–150. ISBN: 0-7695-1893-1. DOI: 10.1109/PERCOM.20

03.1192736. URL: http://ieeexplore.ieee.org/document/1192736/ (cit. on p. 47).

[You04] M. Youssef. "HORUS: A WLAN-based indoor location determination system". PhD Dissertation. University of Maryland, 2004. URL: https://drum.lib.umd.edu/handle/1903/1364 (cit. on p. 54).

[YA04] M. Youssef and A. Agrawala. "Handling samples correlation in the Horus system". In: *IEEE INFOCOM 2004*. Vol. 2. IEEE, 2004, pp. 1023–1031. ISBN: 0-7803-8355-9. DOI: 10.1109/INFCOM.2004.1356989. URL: http://ieeexplore.ieee.org/document/1356989/ (cit. on pp. 47, 54).

[YA05] M. Youssef and A. Agrawala. "The Horus WLAN location determination system". In: *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*. New York, New York, USA: ACM Press, 2005, p. 205. ISBN: 1931971315. DOI: 10.1145/1067170.1067193. URL: http://portal.acm.org/citation.cfm?doid=1067170.1067193 (cit. on p. 54).

[YRN19] T. Yu, W. Ren, and K. Nahrstedt. "MMLOC: multi-mode indoor localization system based on smart access points". In: *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (2019-11), pp. 473–482. DOI: 10.1145/3360774.3360776. URL: https://dl.acm.org/doi/10.1145/3360774.3360776 (cit. on p. 55).

[Zaf+17] F. Zafari et al. "Enhancing the accuracy of iBeacons for indoor proximity-based services". In: *2017 IEEE International Conference on Communications (ICC)* (2017-05), pp. 1–7. ISSN: 15503607. DOI: 10.1109/ICC.2017.7996508. URL: http://ieeexplore.ieee.org/document/7996508/ (cit. on pp. 41, 51, 59).

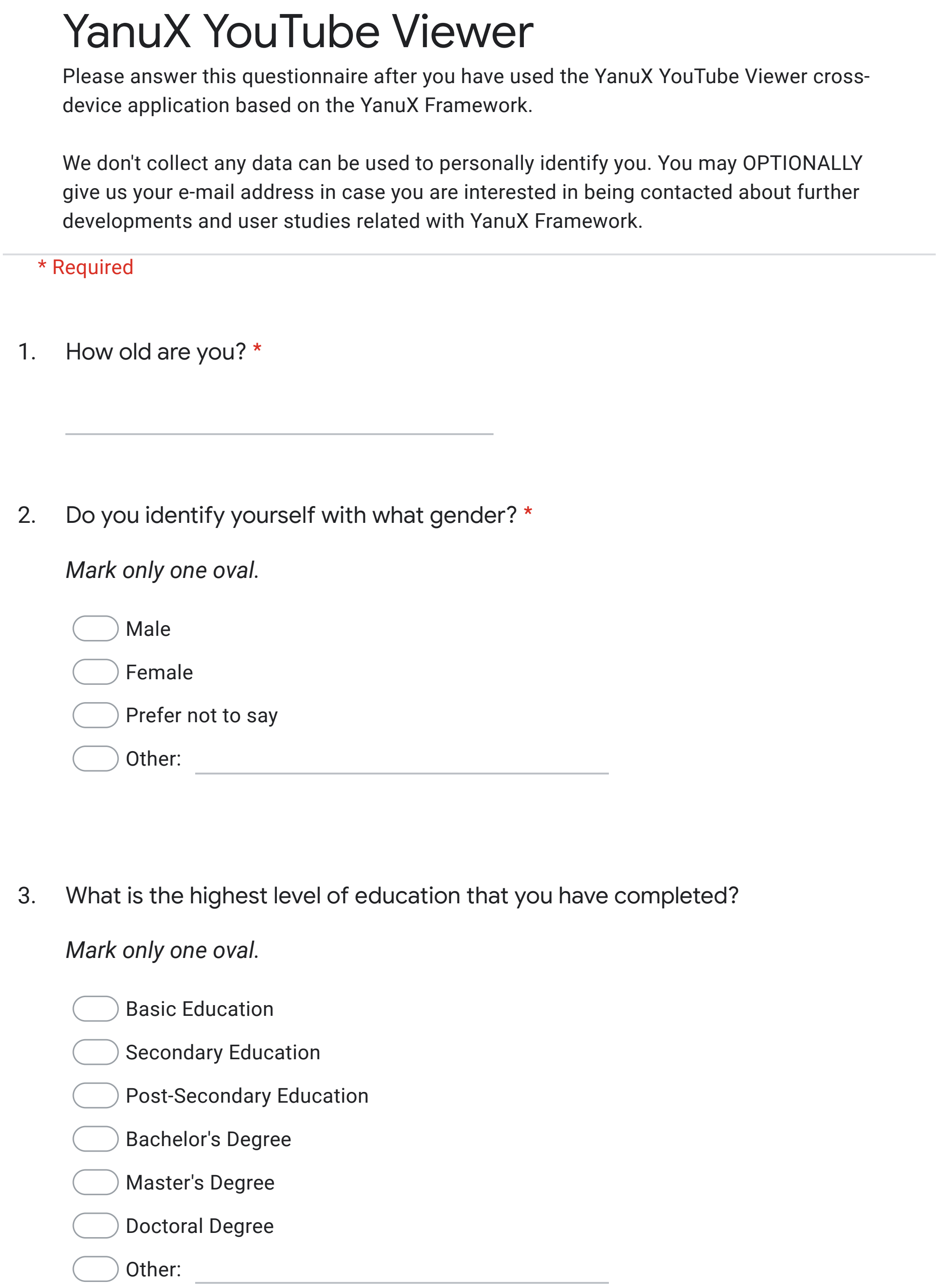
[ZCK14] W. Zhang, M. I. S. Chowdhury, and M. Kavehrad. "Asynchronous indoor positioning system based on visible light communications". In: *Optical Engineering* 53.4 (2014-04), p. 045105. ISSN: 0091-3286. DOI: 10.1117/1.OE.53.4.045105. URL: http://opticalengineering.spiedigitallibrary.org/article.aspx?doi=10.1117/1.OE.53.4.045105 (cit. on p. 61).

[Zho+18] M. Zhou et al. "Fast Fingerprint Database Construction Method in Bluetooth Indoor Positioning System". In: *Proceedings of the 11th EAI International Conference on Mobile Multimedia Communications*. Vol. 2018-June. EAI, 2018-09. ISBN: 978-1-63190-164-5. DOI: 10.4108/eai.21-6-2018.2276621. URL: http://eudl.eu/doi/10.4108/eai.21-6-2018.2276621 (cit. on p. 56).

[Zhu+97] C. Zhu et al. "Algorithm 778: L-BFGS-B". In: *ACM Transactions on Mathematical Software* 23.4 (1997-12), pp. 550–560. ISSN: 0098-3500. DOI: 10.1145/279232.279236. URL: https://doi.org/10.1145/279232.279236%20https://dl.acm.org/doi/10.1145/279232.279236 (cit. on p. 118).

A

YanuX YouTube Viewer Questionnaire



YanuX YouTube Viewer

Please answer this questionnaire after you have used the YanuX YouTube Viewer cross-device application based on the YanuX Framework.

We don't collect any data can be used to personally identify you. You may OPTIONALLY give us your e-mail address in case you are interested in being contacted about further developments and user studies related with YanuX Framework.

* Required

1. How old are you? *

   ____

2. Do you identify yourself with what gender? *

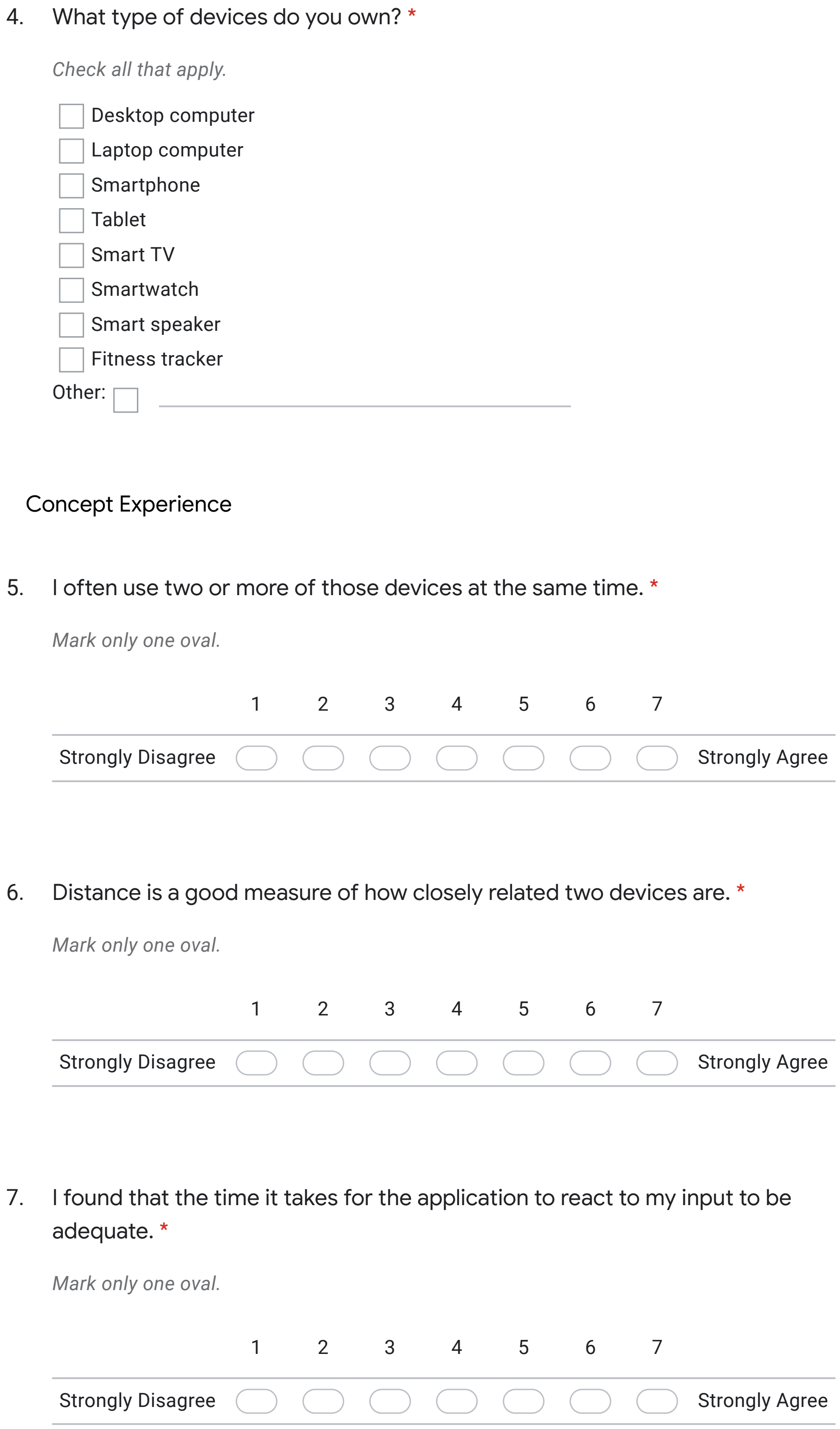
   Mark only one oval.

   ○ Male
   ○ Female
   ○ Prefer not to say
   ○ Other: ____________________

3. What is the highest level of education that you have completed?

   Mark only one oval.

   ○ Basic Education
   ○ Secondary Education
   ○ Post-Secondary Education
   ○ Bachelor's Degree
   ○ Master's Degree
   ○ Doctoral Degree
   ○ Other: ____________________



4. What type of devices do you own? *

Check all that apply.

□ Desktop computer
□ Laptop computer
□ Smartphone
□ Tablet
□ Smart TV
□ Smartwatch
□ Smart speaker
□ Fitness tracker
Other: □ ____________________

Concept Experience

5. I often use two or more of those devices at the same time. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

6. Distance is a good measure of how closely related two devices are. *

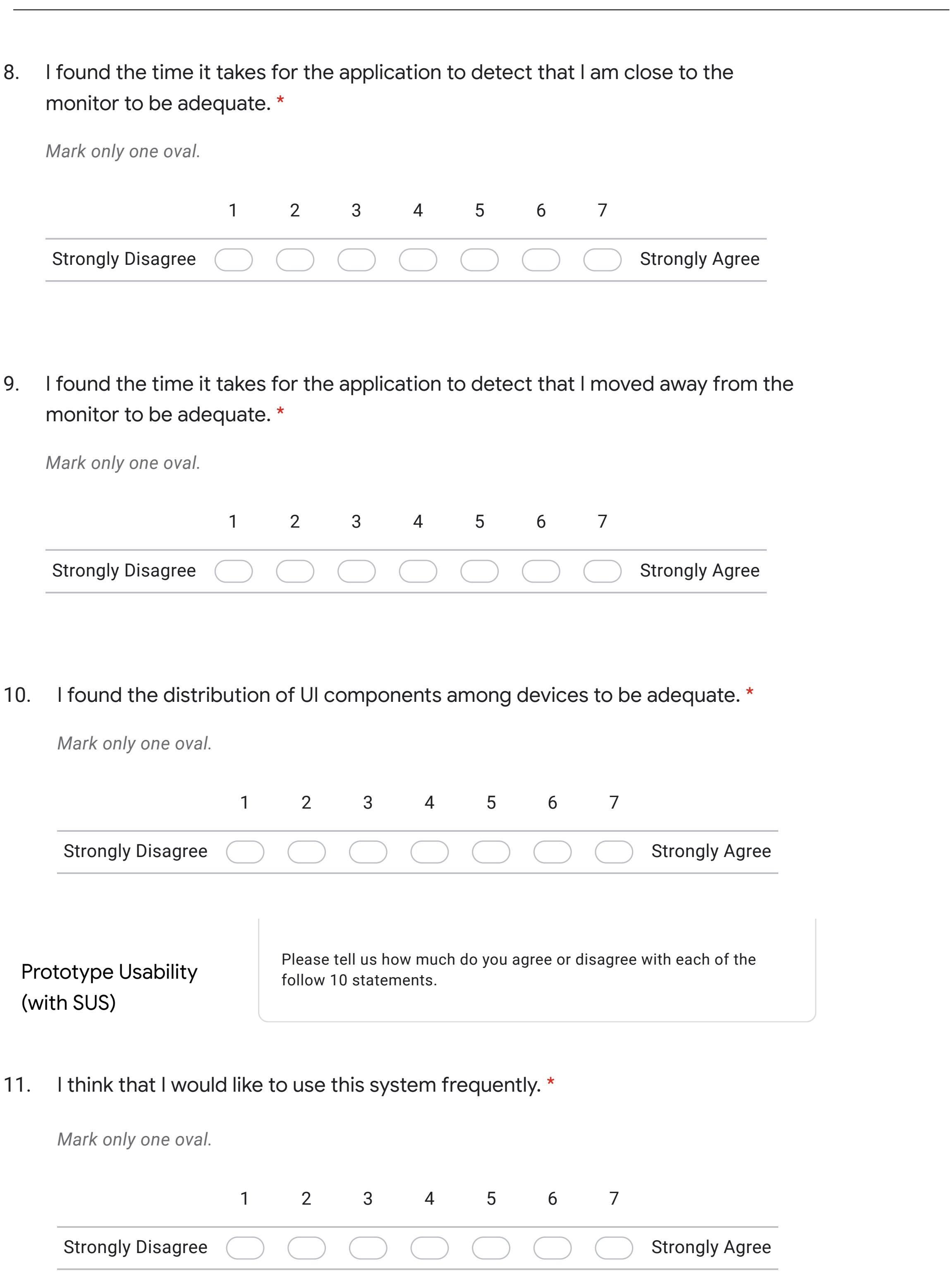
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

7. I found that the time it takes for the application to react to my input to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



8. I found the time it takes for the application to detect that I am close to the monitor to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

9. I found the time it takes for the application to detect that I moved away from the monitor to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

10. I found the distribution of UI components among devices to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

Prototype Usability (with SUS)

Please tell us how much do you agree or disagree with each of the follow 10 statements.

11. I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

12. I found the system unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

13. I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

14. I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

15. I found the various functions in this system were well integrated. *

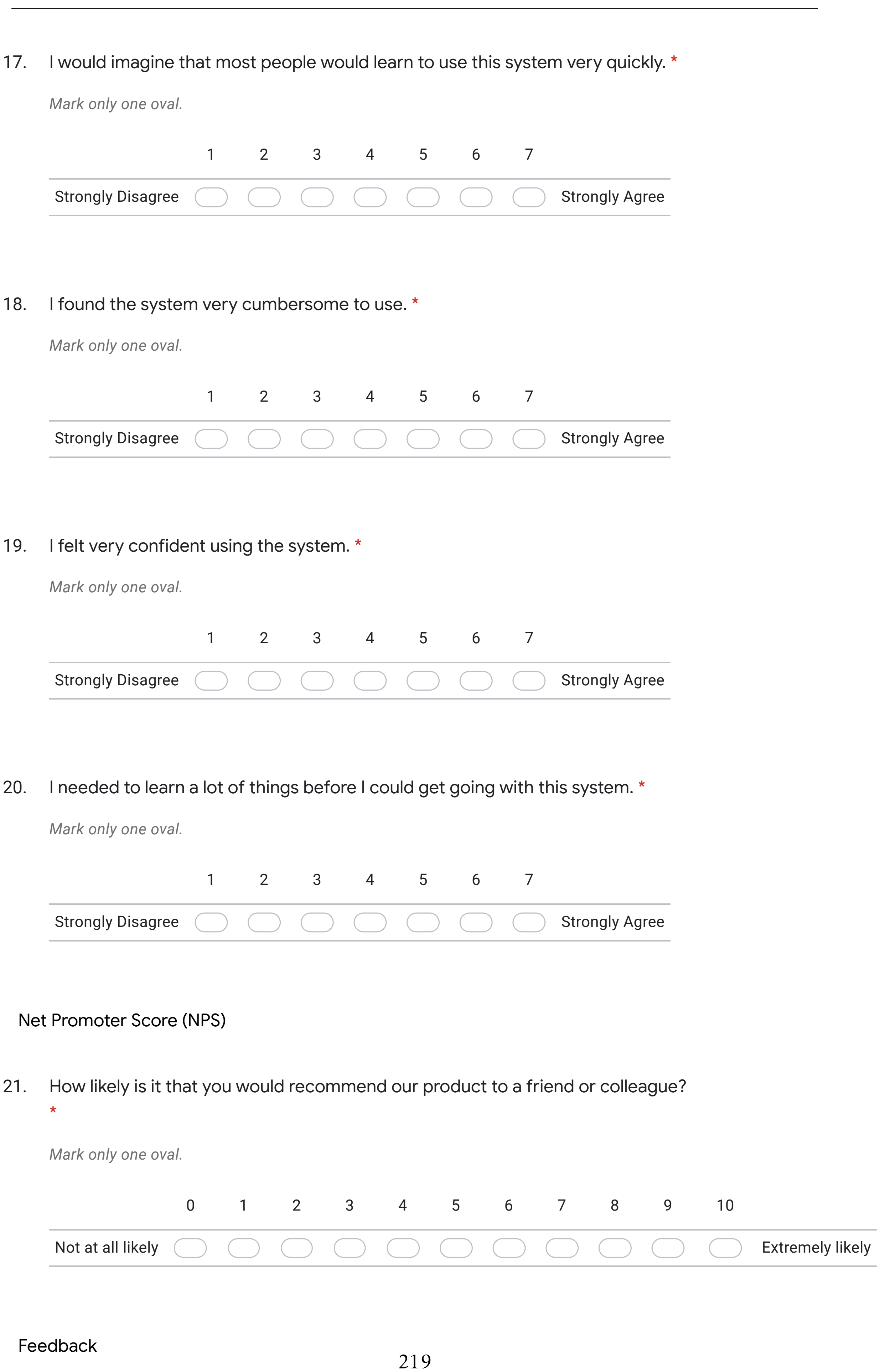
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

16. I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



17. I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

18. I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

19. I felt very confident using the system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

20. I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

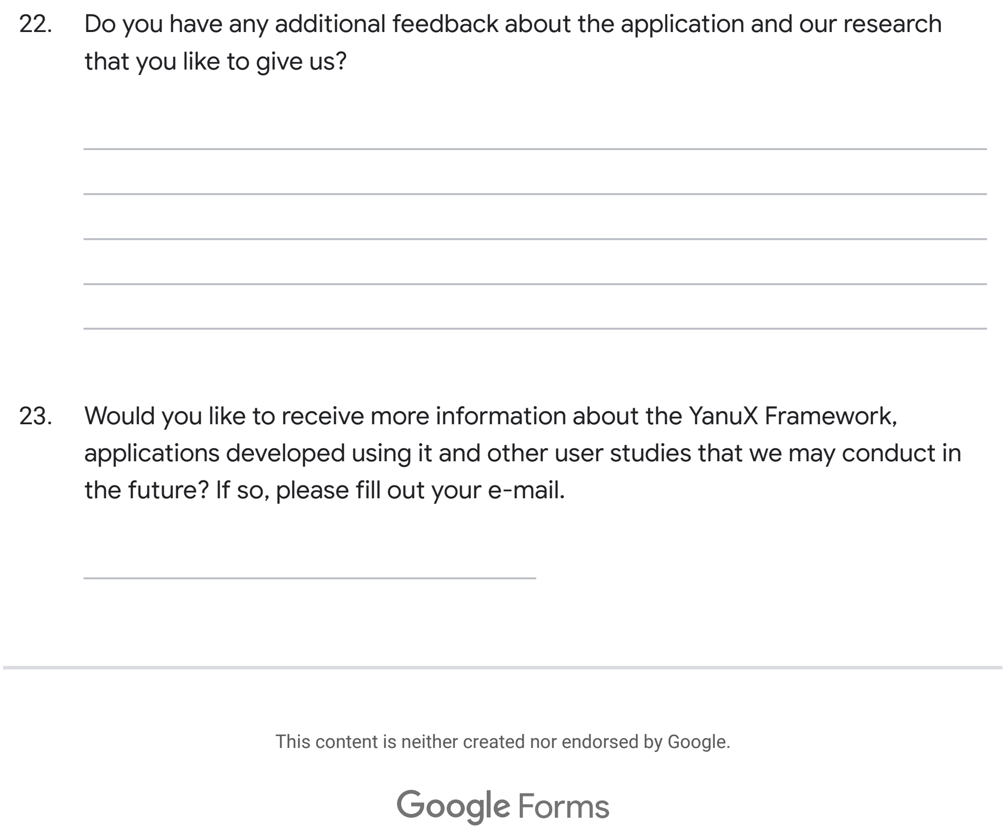
Net Promoter Score (NPS)

21. How likely is it that you would recommend our product to a friend or colleague? *

Mark only one oval.

	0	1	2	3	4	5	6	7	8	9	10	
Not at all likely	○	○	○	○	○	○	○	○	○	○	○	Extremely likely

Feedback

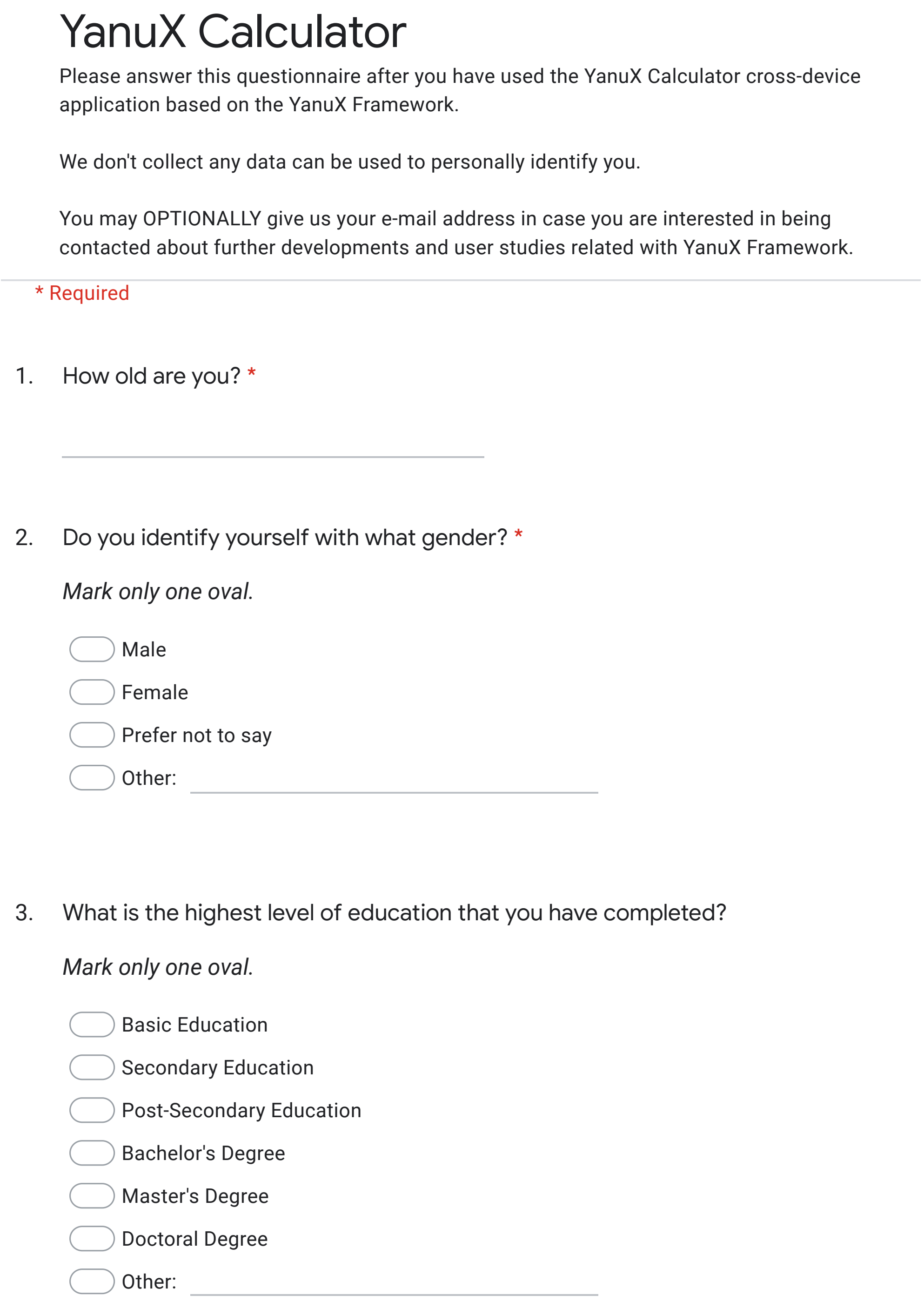


22. Do you have any additional feedback about the application and our research that you like to give us?

23. Would you like to receive more information about the YanuX Framework, applications developed using it and other user studies that we may conduct in the future? If so, please fill out your e-mail.

B

YanuX Calculator Questionnaire



YanuX Calculator

Please answer this questionnaire after you have used the YanuX Calculator cross-device application based on the YanuX Framework.

We don't collect any data can be used to personally identify you.

You may OPTIONALLY give us your e-mail address in case you are interested in being contacted about further developments and user studies related with YanuX Framework.

* Required

1. How old are you? *

____

2. Do you identify yourself with what gender? *

Mark only one oval.

○ Male

○ Female

○ Prefer not to say

○ Other: ____________________________

3. What is the highest level of education that you have completed?

Mark only one oval.

○ Basic Education

○ Secondary Education

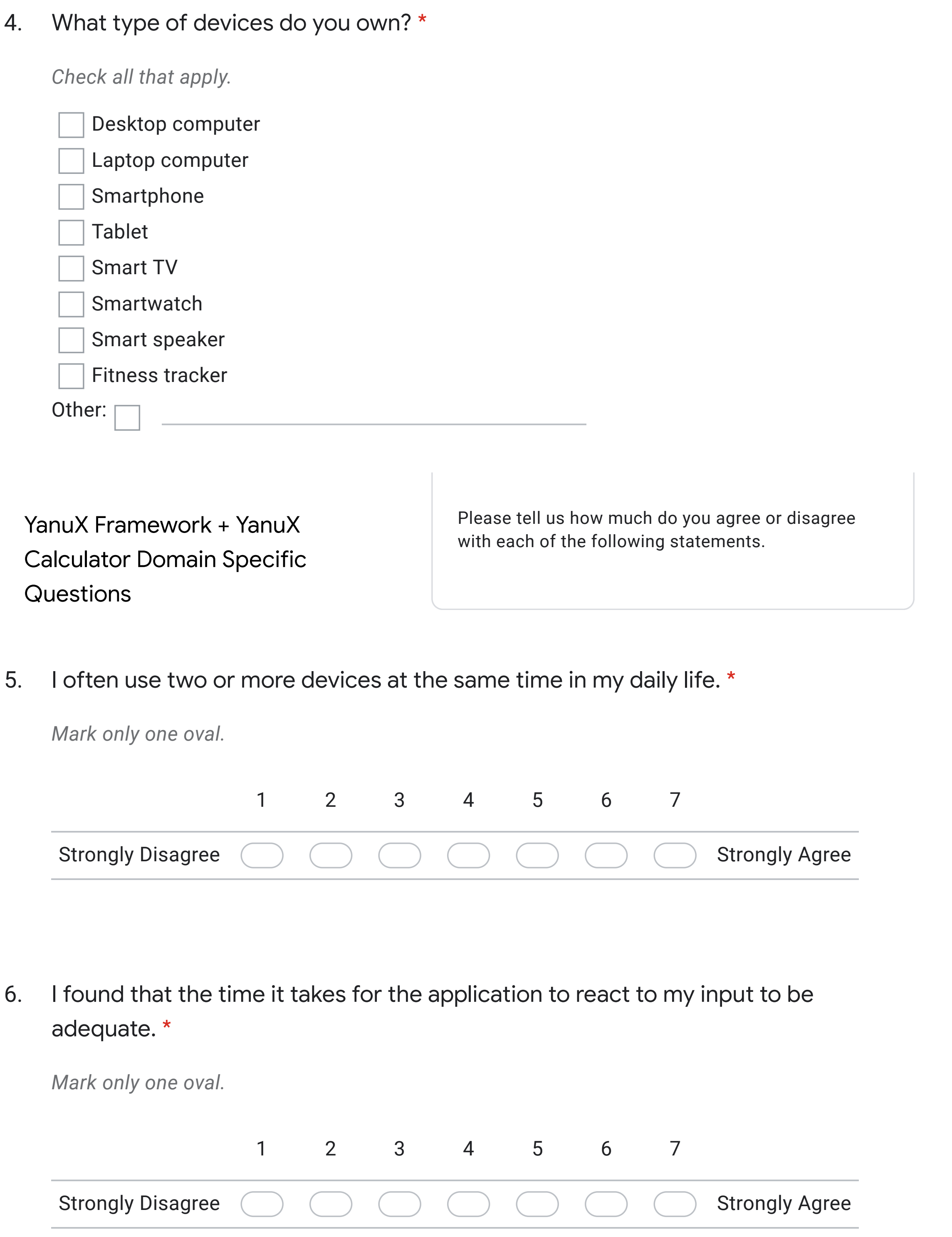
○ Post-Secondary Education

○ Bachelor's Degree

○ Master's Degree

○ Doctoral Degree

○ Other: ____________________________



4. What type of devices do you own? *

Check all that apply.

□ Desktop computer
□ Laptop computer
□ Smartphone
□ Tablet
□ Smart TV
□ Smartwatch
□ Smart speaker
□ Fitness tracker
Other: □ ____________________

YanuX Framework + YanuX Calculator Domain Specific Questions

Please tell us how much do you agree or disagree with each of the following statements.

5. I often use two or more devices at the same time in my daily life. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

6. I found that the time it takes for the application to react to my input to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

7. Distance is a good measure of how closely related two devices are. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

8. I found the time it takes for the shared display to react to my presence to be adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

9. I found the automatic distribution of UI components among devices to be adequate. *

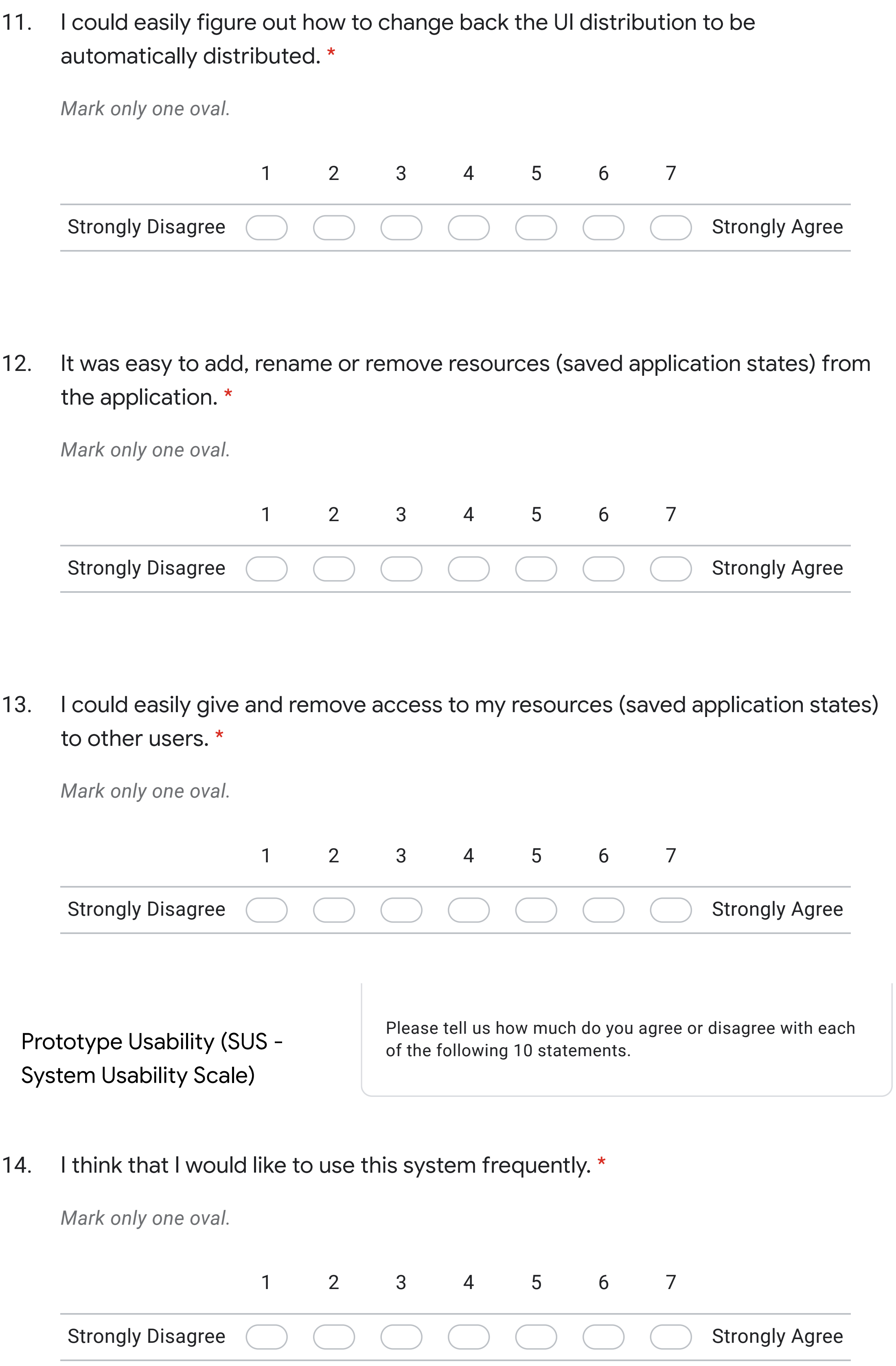
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

10. It is easy to change the distribution of UI components to suit my needs. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



11. I could easily figure out how to change back the UI distribution to be automatically distributed. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

12. It was easy to add, rename or remove resources (saved application states) from the application. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

13. I could easily give and remove access to my resources (saved application states) to other users. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

Prototype Usability (SUS - System Usability Scale)

Please tell us how much do you agree or disagree with each of the following 10 statements.

14. I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

15. I found the system unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

16. I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

17. I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

18. I found the various functions in this system were well integrated. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

19. I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

20. I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

21. I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

22. I felt very confident using the system. *

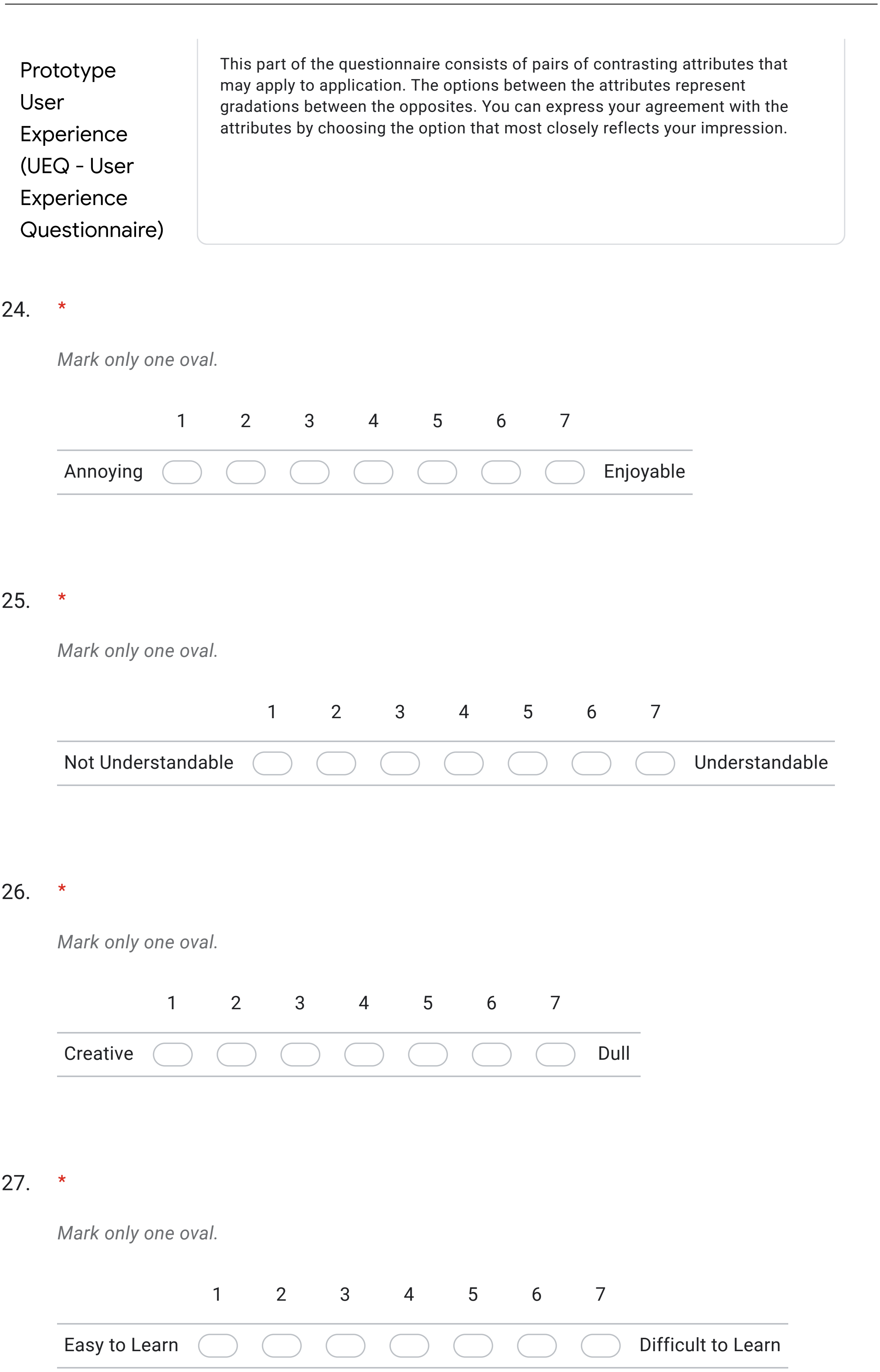
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

23. I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



Prototype User Experience (UEQ - User Experience Questionnaire)

This part of the questionnaire consists of pairs of contrasting attributes that may apply to application. The options between the attributes represent gradations between the opposites. You can express your agreement with the attributes by choosing the option that most closely reflects your impression.

24. *

Mark only one oval.

	1	2	3	4	5	6	7	
Annoying	○	○	○	○	○	○	○	Enjoyable

25. *

Mark only one oval.

	1	2	3	4	5	6	7	
Not Understandable	○	○	○	○	○	○	○	Understandable

26. *

Mark only one oval.

	1	2	3	4	5	6	7	
Creative	○	○	○	○	○	○	○	Dull

27. *

Mark only one oval.

	1	2	3	4	5	6	7	
Easy to Learn	○	○	○	○	○	○	○	Difficult to Learn

28. *

Mark only one oval.

	1	2	3	4	5	6	7	
Valuable	○	○	○	○	○	○	○	Inferior

29. *

Mark only one oval.

	1	2	3	4	5	6	7	
Boring	○	○	○	○	○	○	○	Exciting

30. *

Mark only one oval.

	1	2	3	4	5	6	7	
Not Interesting	○	○	○	○	○	○	○	Interesting

31. *

Mark only one oval.

	1	2	3	4	5	6	7	
Unpredictable	○	○	○	○	○	○	○	Predictable

32. *

Mark only one oval.

	1	2	3	4	5	6	7	
Fast	○	○	○	○	○	○	○	Slow

33. *

Mark only one oval.

	1	2	3	4	5	6	7	
Iventive	○	○	○	○	○	○	○	Conventional

34. *

Mark only one oval.

	1	2	3	4	5	6	7	
Obstructive	○	○	○	○	○	○	○	Supportive

35. *

Mark only one oval.

	1	2	3	4	5	6	7	
Good	○	○	○	○	○	○	○	Bad

36. *

Mark only one oval.

	1	2	3	4	5	6	7	
Complicated	○	○	○	○	○	○	○	Easy

37. *

Mark only one oval.

	1	2	3	4	5	6	7	
Unlikable	○	○	○	○	○	○	○	Pleasing

38. *

Mark only one oval.

	1	2	3	4	5	6	7	
Usual	○	○	○	○	○	○	○	Leading Edge

39. *

Mark only one oval.

	1	2	3	4	5	6	7	
Unpleasant	○	○	○	○	○	○	○	Pleasant

40. *

Mark only one oval.

	1	2	3	4	5	6	7	
Secure	○	○	○	○	○	○	○	Not Secure

41. *

Mark only one oval.

	1	2	3	4	5	6	7	
Motivating	○	○	○	○	○	○	○	Demotivating

42. *

Mark only one oval.

	1	2	3	4	5	6	7	
Meets Expectations	○	○	○	○	○	○	○	Does Not Meet Expectations

43. *

Mark only one oval.

	1	2	3	4	5	6	7	
Inefficient	○	○	○	○	○	○	○	Efficient

44. *

Mark only one oval.

	1	2	3	4	5	6	7	
Clear	○	○	○	○	○	○	○	Confusing

45. *

Mark only one oval.

	1	2	3	4	5	6	7	
Impractical	○	○	○	○	○	○	○	Practical

46. *

Mark only one oval.

	1	2	3	4	5	6	7	
Organized	○	○	○	○	○	○	○	Cluttered

47. *

Mark only one oval.

	1	2	3	4	5	6	7	
Attractive	○	○	○	○	○	○	○	Unattractive

48. *

Mark only one oval.

	1	2	3	4	5	6	7	
Friendly	○	○	○	○	○	○	○	Unfriendly

49. *

Mark only one oval.

	1	2	3	4	5	6	7	
Conservative	○	○	○	○	○	○	○	Innovative

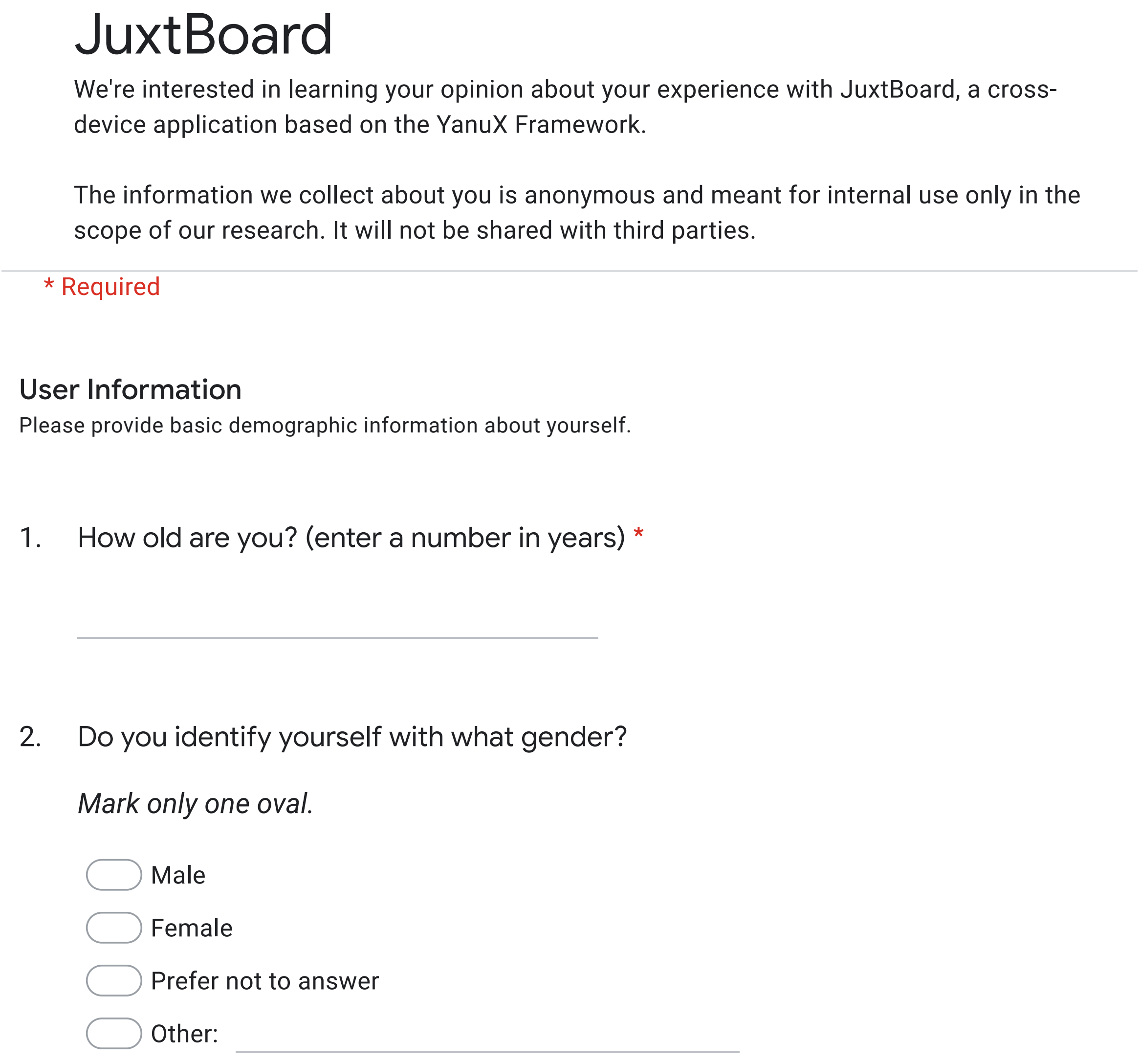
Feedback

50. Do you have any additional feedback about the application and our research that you like to give us?

51. Would you like to receive more information about the YanuX Framework, applications developed using it and other user studies that we may conduct in the future? If so, please fill out your e-mail.

C

JuxtBoard Questionnaire



JuxtBoard

We're interested in learning your opinion about your experience with JuxtBoard, a cross-device application based on the YanuX Framework.

The information we collect about you is anonymous and meant for internal use only in the scope of our research. It will not be shared with third parties.

* Required

User Information

Please provide basic demographic information about yourself.

1. How old are you? (enter a number in years) *

____

2. Do you identify yourself with what gender?

Mark only one oval.

○ Male

○ Female

○ Prefer not to answer

○ Other: ______________________________

3. What is the highest degree or level of school you have completed? *

Mark only one oval.

○ Basic Education
○ Secondary Education
○ Post-Secondary Education
○ Bachelor's Degree
○ Master's Degree
○ Doctoral Degree
○ Prefer not to answer
○ Other: ____________________

4. What type of computational devices do you own? *

Check all that apply.

□ Desktop computer
□ Laptop computer
□ Smartphone
□ Tablet
□ Smart TV
□ Smartwatch
□ Smart speaker
□ Fitness tracker
□ None
Other: □ ____________________

5. I often use two or more devices at the same time in my daily life. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

6. I was asked to use JuxtBoard ... *

Mark only one oval.

○ FIRST in Single Device Mode and THEN in Cross-device Mode

○ FIRST in Cross-device Mode and THEN in Single Device Mode

JuxtBoard Single Device Interaction Mode

Recall your experience with JuxtBoard in "Single Device" mode and please tell us how much do you agree or disagree with each of the following statements.

7. I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

8. I found the system unnecessarily complex. *

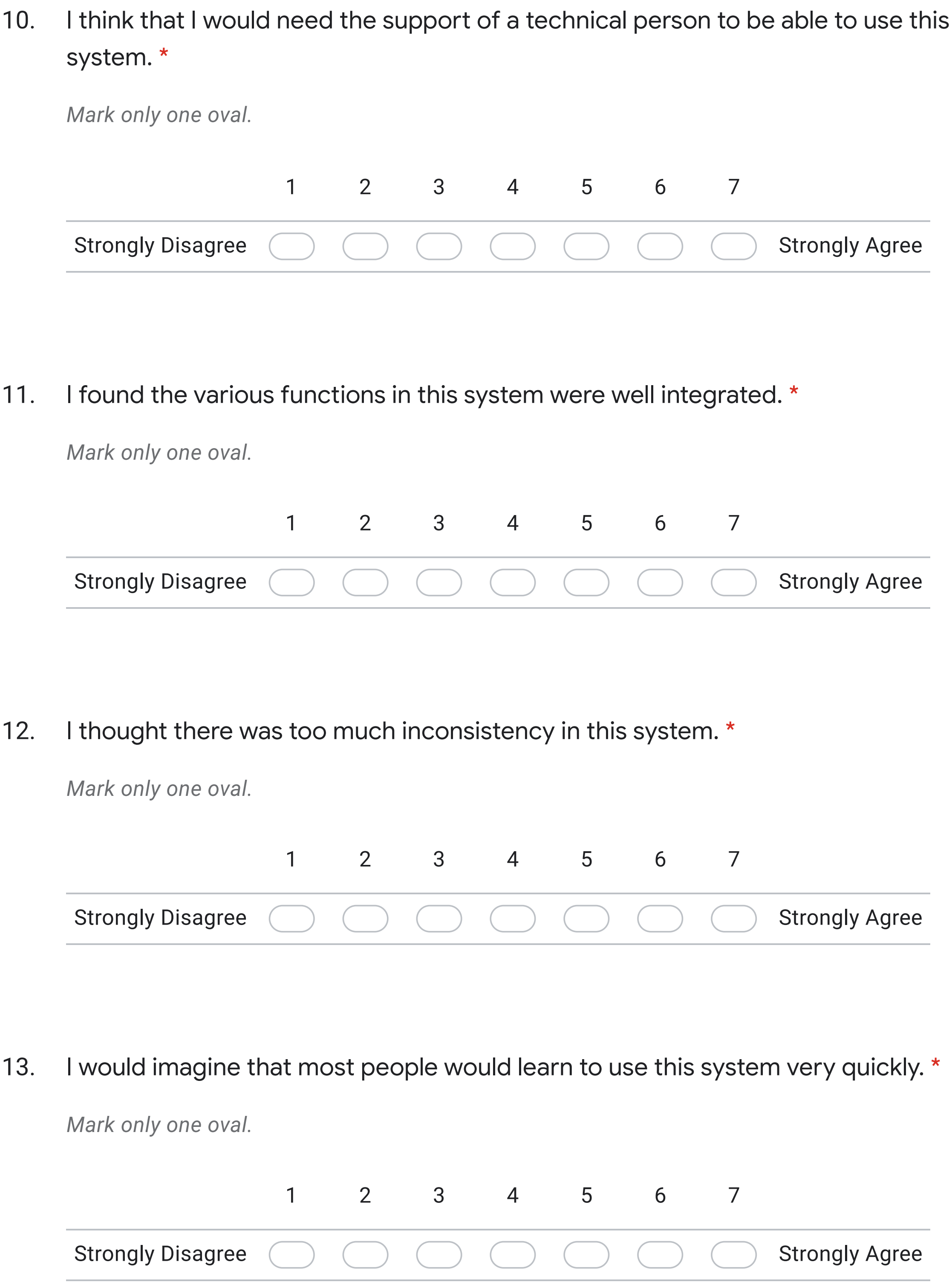
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

9. I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



10. I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

11. I found the various functions in this system were well integrated. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

12. I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

13. I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

14. I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

15. I felt very confident using the system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

16. I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

System Usability Scale - JuxtBoard Cross-device Interaction Mode

Recall your experience with JuxtBoard in "Cross-device" mode and please tell us how much do you agree or disagree with each of the following statements.

17. I think that I would like to use this system frequently. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

18. I found the system unnecessarily complex. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

19. I thought the system was easy to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

20. I think that I would need the support of a technical person to be able to use this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

21. I found the various functions in this system were well integrated. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

22. I thought there was too much inconsistency in this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

23. I would imagine that most people would learn to use this system very quickly. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

24. I found the system very cumbersome to use. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

25. I felt very confident using the system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

26. I needed to learn a lot of things before I could get going with this system. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

Single Device vs. Cross-device

Please indicate your preference towards each of the interaction models (Single Device or Cross-device Mode) based on your experience while using JuxtBoard.

Please note that 1 means that you found Single Device to absolutely be the most adequate/preferred mode for a certain situation or task. 7 means the same for the Cross-device Mode. Intermediate values should indicate intermediate levels of adequacy/preference between the two interaction modes, with 4 meaning that neither of the two modes is the most adequate/preferred.

27. In which mode was it easier to manage the collections of notes in the application (e.g., selecting, adding, sharing, renaming or removing collections of notes)? *

Mark only one oval.

	1	2	3	4	5	6	7	
Single Device	○	○	○	○	○	○	○	Cross-device

28. In which mode was it easier to manage the notes in the application (e.g., selecting, adding, renaming or removing notes)? *

Mark only one oval.

	1	2	3	4	5	6	7	
Single Device	○	○	○	○	○	○	○	Cross-device

29. In which mode was it easier to collaborate and show notes to other users? *

Mark only one oval.

	1	2	3	4	5	6	7	
Single Device	○	○	○	○	○	○	○	Cross-device

30. What is your degree of preference between the two modes when using the application to perform tasks alone? *

Mark only one oval.

	1	2	3	4	5	6	7	
Single Device	○	○	○	○	○	○	○	Cross-device

31. What is your degree of preference between the two modes when using the application to perform tasks in collaboration with other users? *

Mark only one oval.

	1	2	3	4	5	6	7	
Single Device	○	○	○	○	○	○	○	Cross-device

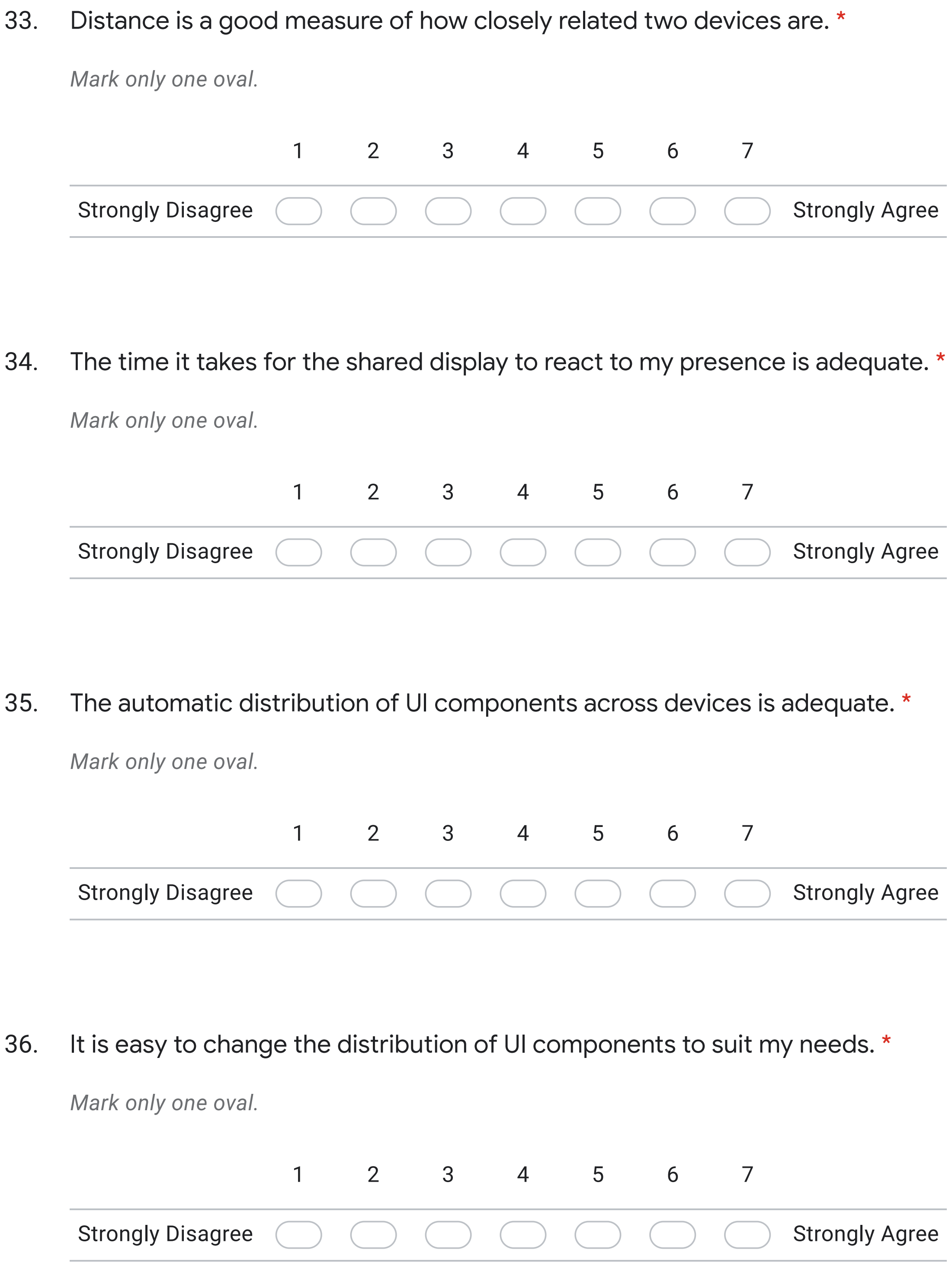
Utilização Geral

Com base na sua experiência de utilização do JuxtBoard, indique o nível de concordância com cada uma das situações apresentadas.

32. The time it takes for the application to react to my input is adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



33. Distance is a good measure of how closely related two devices are. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

34. The time it takes for the shared display to react to my presence is adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

35. The automatic distribution of UI components across devices is adequate. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

36. It is easy to change the distribution of UI components to suit my needs. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

37. It was easy to add, rename, remove, share and unshare collections of notes in the application. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

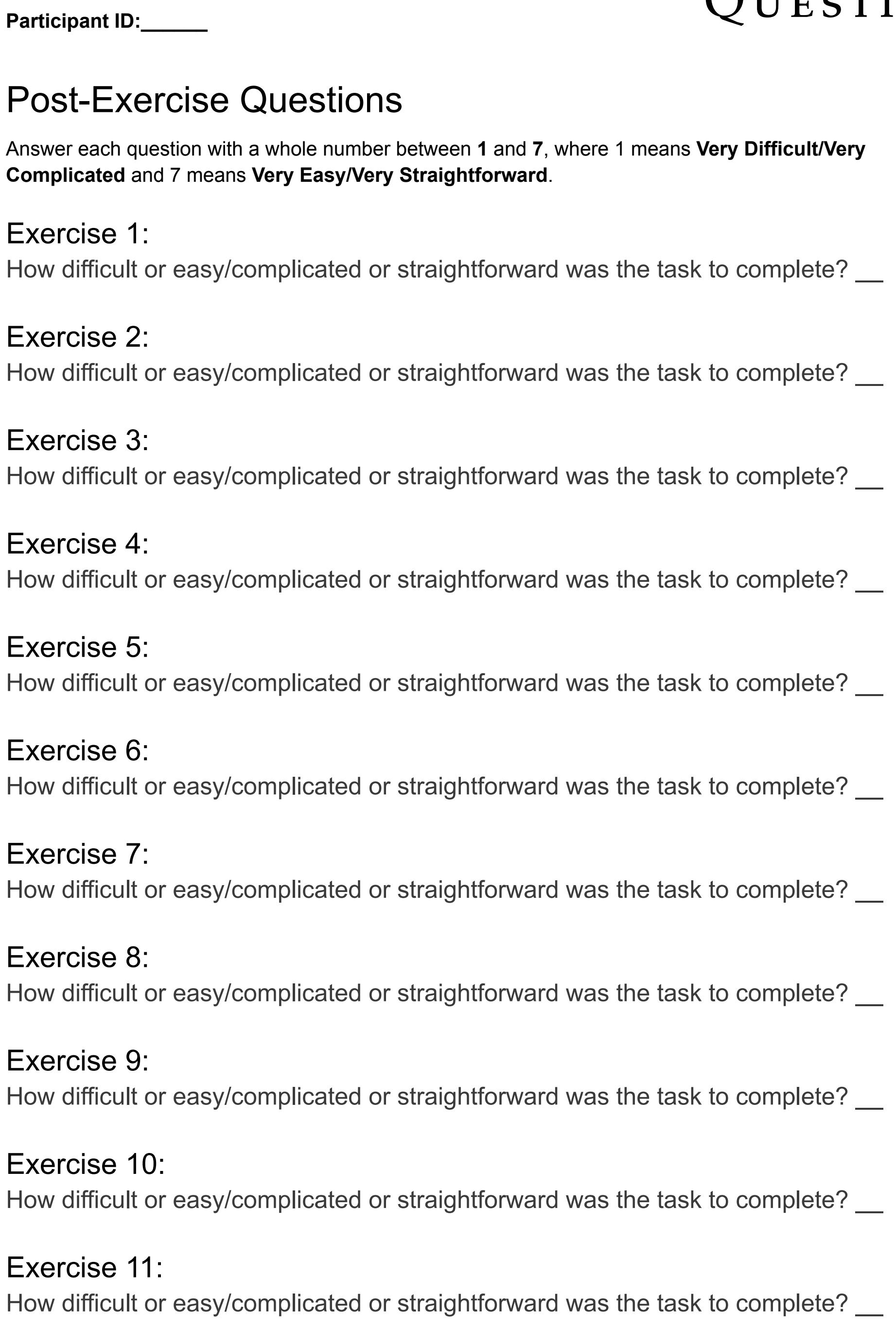
Feedback

Provide additional feedback about your experience with JuxtBoard.

38. Do you have additional feedback about your experience with JuxtBoard?

D

YanuX Skeletron Post-Exercise Questions



Participant ID:______

Post-Exercise Questions

Answer each question with a whole number between **1** and **7**, where 1 means **Very Difficult/Very Complicated** and 7 means **Very Easy/Very Straightforward**.

Exercise 1:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 2:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 3:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 4:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 5:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 6:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 7:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 8:
How difficult or easy/complicated or straightforward was the task to complete? __

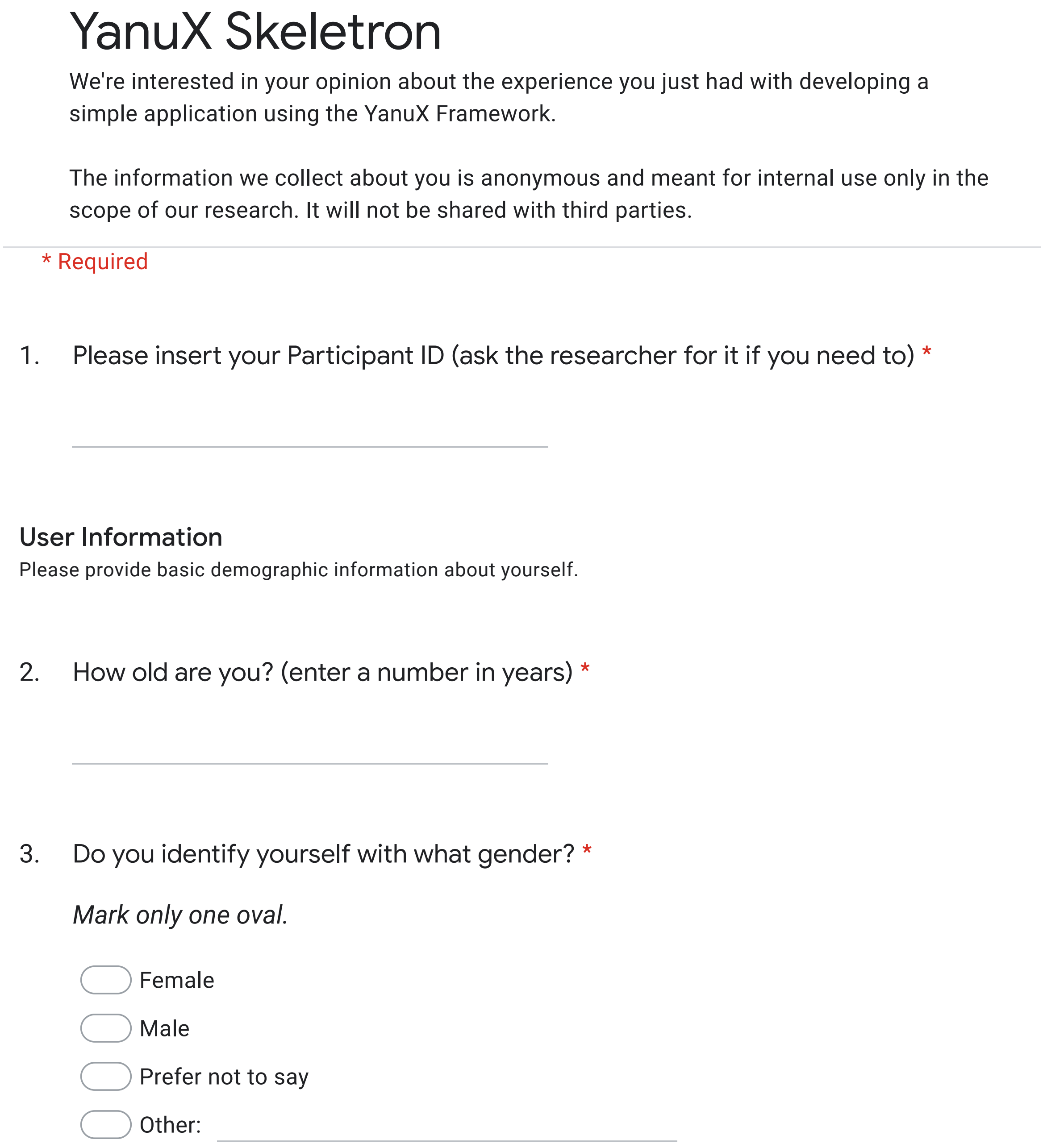
Exercise 9:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 10:
How difficult or easy/complicated or straightforward was the task to complete? __

Exercise 11:
How difficult or easy/complicated or straightforward was the task to complete? __

E

JuxtBoard Questionnaire



YanuX Skeletron

We're interested in your opinion about the experience you just had with developing a simple application using the YanuX Framework.

The information we collect about you is anonymous and meant for internal use only in the scope of our research. It will not be shared with third parties.

* Required

1. Please insert your Participant ID (ask the researcher for it if you need to) *

   ____

User Information

Please provide basic demographic information about yourself.

2. How old are you? (enter a number in years) *

   ____

3. Do you identify yourself with what gender? *

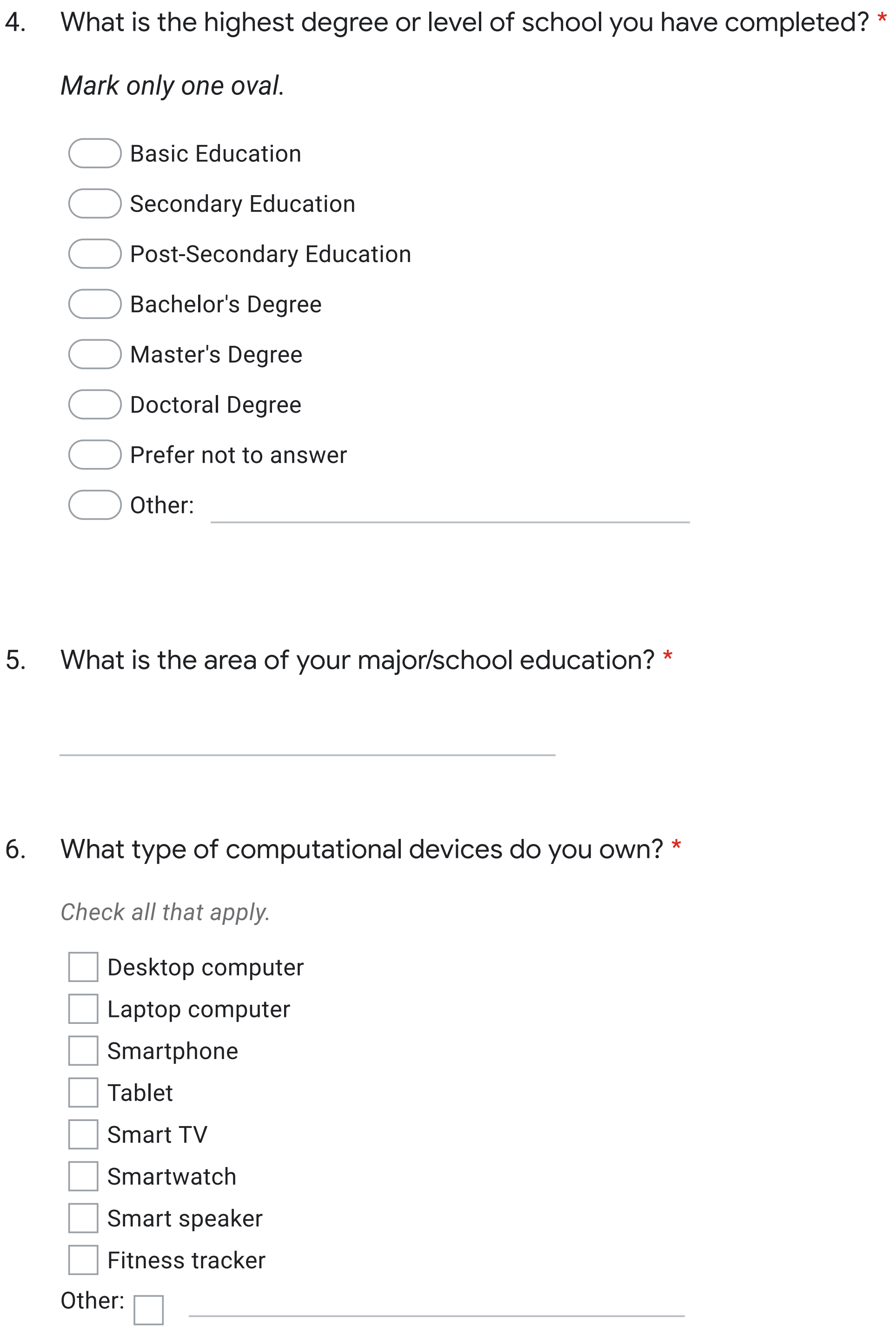
   Mark only only one oval.

   ○ Female

   ○ Male

   ○ Prefer not to say

   ○ Other: ______________________________



4. What is the highest degree or level of school you have completed? *

Mark only one oval.

○ Basic Education

○ Secondary Education

○ Post-Secondary Education

○ Bachelor's Degree

○ Master's Degree

○ Doctoral Degree

○ Prefer not to answer

○ Other: ______________________

5. What is the area of your major/school education? *

____

6. What type of computational devices do you own? *

Check all that apply.

□ Desktop computer

□ Laptop computer

□ Smartphone

□ Tablet

□ Smart TV

□ Smartwatch

□ Smart speaker

□ Fitness tracker

Other: □ ______________________

7. I often use two or more devices at the same time in my daily life. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

Programming Experience

Answer to the following questions regarding your programming experience.

8. For how many years have you been programming? *

____

9. How experienced are you with programming? *

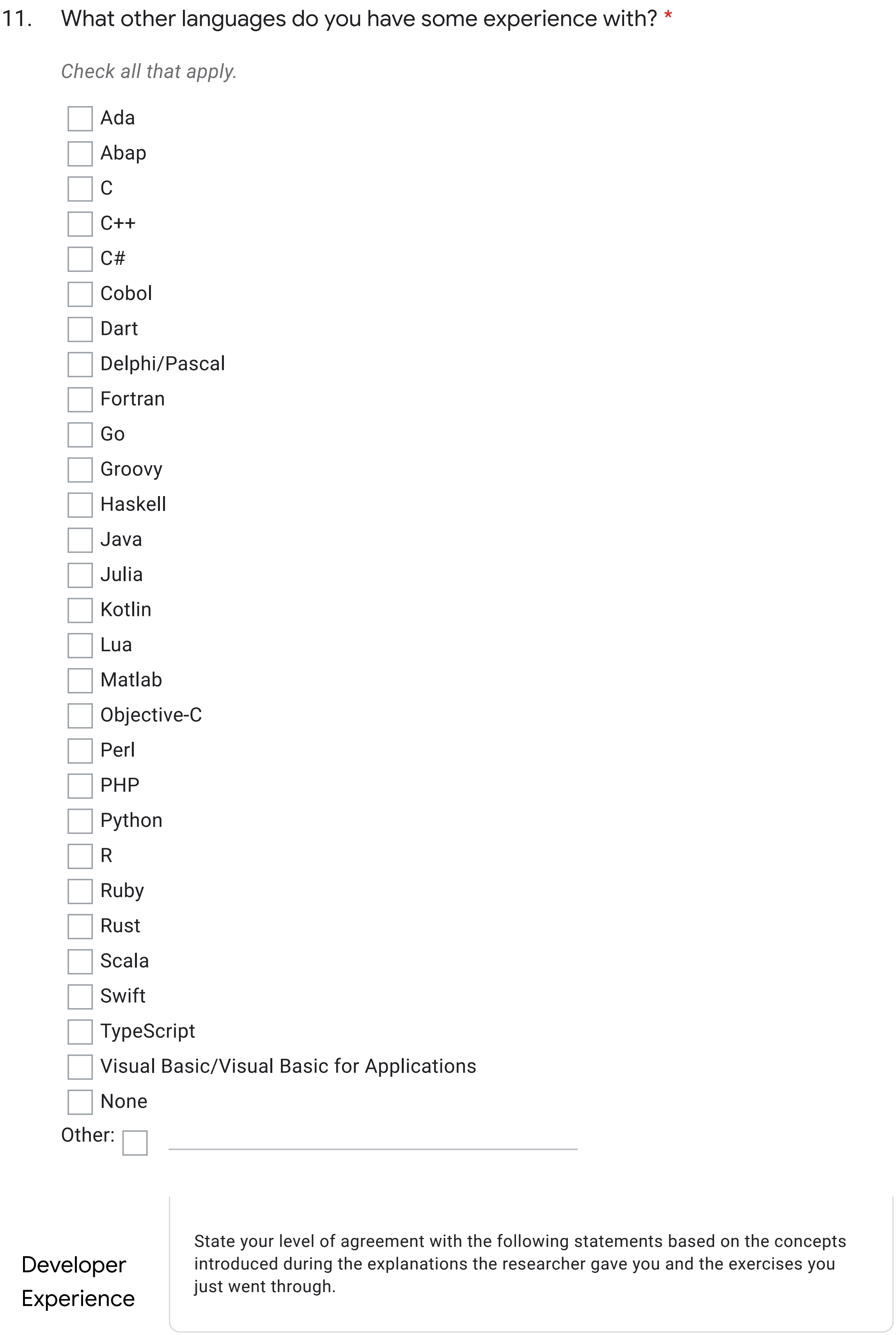
Mark only one oval.

	1	2	3	4	5	6	7	
Beginner	○	○	○	○	○	○	○	Expert

10. How experienced are you with JavaScript? *

Mark only one oval.

	1	2	3	4	5	6	7	
Beginner	○	○	○	○	○	○	○	Expert



11. What other languages do you have some experience with? *

Check all that apply.

□ Ada
□ Abap
□ C
□ C++
□ C#
□ Cobol
□ Dart
□ Delphi/Pascal
□ Fortran
□ Go
□ Groovy
□ Haskell
□ Java
□ Julia
□ Kotlin
□ Lua
□ Matlab
□ Objective-C
□ Perl
□ PHP
□ Python
□ R
□ Ruby
□ Rust
□ Scala
□ Swift
□ TypeScript
□ Visual Basic/Visual Basic for Applications
□ None

Other: □ ______________________________

Developer Experience

State your level of agreement with the following statements based on the concepts introduced during the explanations the researcher gave you and the exercises you just went through.

12. It makes sense that the UI state of an application is represented by an object. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

13. I understood how to save the UI state of an application every time it changes. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

14. I understood how to use the "Coordinator" API to perform various tasks that make my cross-device application behave properly? *

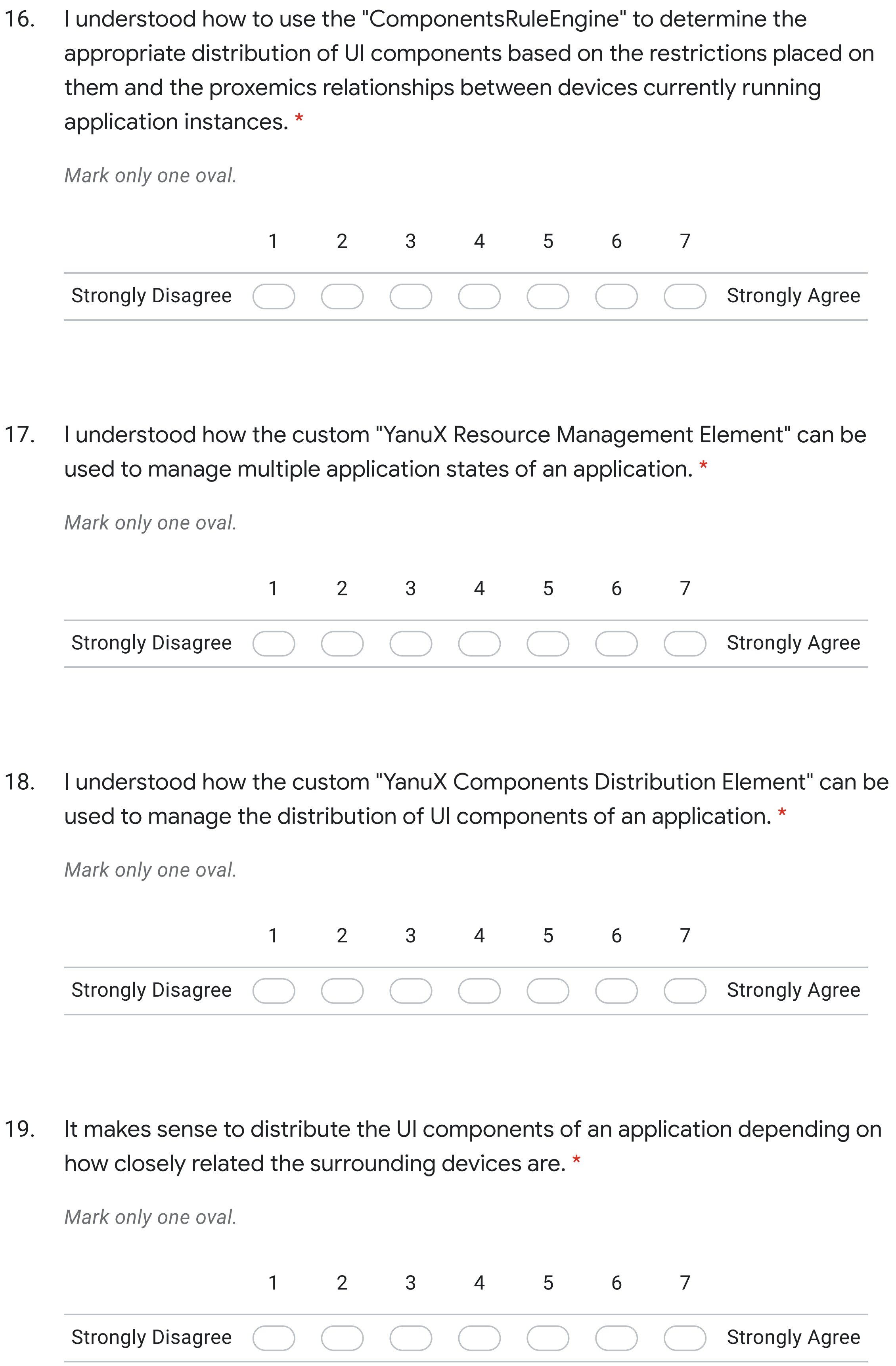
Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

15. I understood how the DSL (Domain Specific Language) for the automatic distribution of UI components works. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree



16. I understood how to use the "ComponentsRuleEngine" to determine the appropriate distribution of UI components based on the restrictions placed on them and the proxemics relationships between devices currently running application instances. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

17. I understood how the custom "YanuX Resource Management Element" can be used to manage multiple application states of an application. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

18. I understood how the custom "YanuX Components Distribution Element" can be used to manage the distribution of UI components of an application. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

19. It makes sense to distribute the UI components of an application depending on how closely related the surrounding devices are. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

20. The YanuX Framework's capabilities meet my requirements. *

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

21. The YanuX Framework is easy to use.

Mark only one oval.

	1	2	3	4	5	6	7	
Strongly Disagree	○	○	○	○	○	○	○	Strongly Agree

Semantic Differential Scale

How do you evaluate the YanuX Framework?
(Choose between the level of agreement between the two the extreme points)

22. *

Mark only one oval.

	1	2	3	4	5	6	7	
Uninteresting	○	○	○	○	○	○	○	Interesting

23. *

Mark only one oval.

	1	2	3	4	5	6	7	
Complex	○	○	○	○	○	○	○	Simple

24. *

Mark only one oval.

	1	2	3	4	5	6	7	
Inappropriate	○	○	○	○	○	○	○	Appropriate

25. *

Mark only one oval.

	1	2	3	4	5	6	7	
Hard to Understand	○	○	○	○	○	○	○	Easy to Understand

26. *

Mark only one oval.

	1	2	3	4	5	6	7	
Hard to Learn	○	○	○	○	○	○	○	Easy to Learn

27. *

Mark only one oval.

	1	2	3	4	5	6	7	
Hard to Develop	○	○	○	○	○	○	○	Easy to Develop

Recomendation

Helps us understand your opinion about the "YanuX Framework".

28. How likely is that you would recommend the YanuX Framework to other developers you known? *

Mark only one oval.

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

NASA Task Load Index

The NASA Task Load Index should be answered on a separate sheet.

Feedback

Provide additional feedback about your experience with the YanuX Framework.

29. Do you have additional feedback about your experience with the YanuX Framework?

30. If you intend to receive any updates about the YanuX Framework, please provide use your e-mail address. Rest assured that it will not be used for any other purpose.

NASA Task Load Index

Rating Scale Definitions

Mental Demand

How much mental and perceptual activity was required (e g . thinking. deciding. calculating. remembering. looking. searching. etc. Was the task easy or demanding. simple or complex. exacting or forgiving?

Physical Demand

How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Temporal Demand

How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Performance

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

Effort

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration

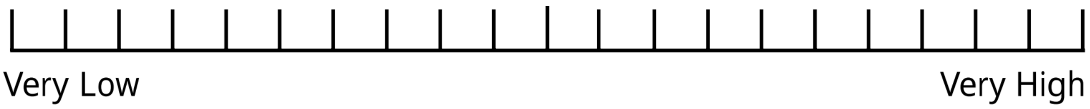
How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?

Participant ID:______

Ratings

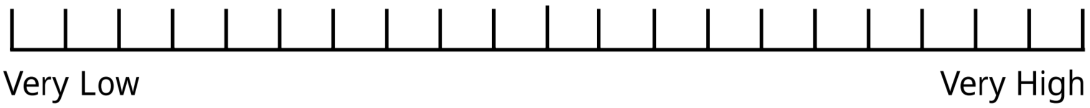
Mental Demand

How mentally demanding was the task?



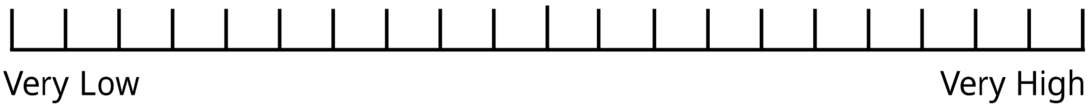
Physical Demand

How physically demanding was the task?



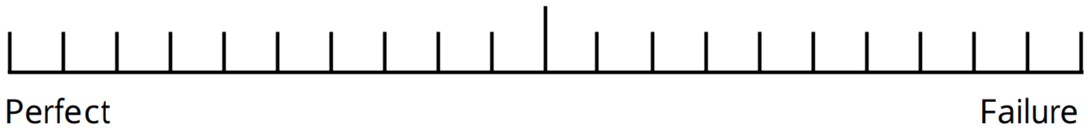
Temporal Demand

How hurried or rushed was the pace of the task?



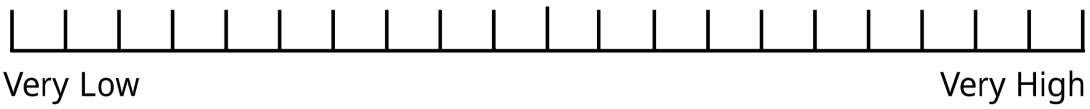
Performance

How successful were you in accomplishing what you were asked to do?



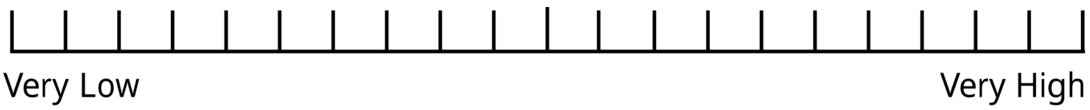
Effort

How hard did you have to work to accomplish your level of performance?



Frustration

How insecure, discouraged, irritated, stressed and annoyed were you?



2021

Applications across Co-located Devices: Studying a Framework-driven Design Approach

Pedro Albuquerque Santos

UNIVERSIDADE NOVA DE LISBOA

OMNIS CIVITAS CONTRA SE DIVISA NON STABIT