Università di Bologna · Universität Bielefeld

Dept. of Computer Science and Engineering · Faculty of Technology

Master's Degree in
Computer Science · Intelligent Interactive Systems

Privacy and Security in Collaborative Mixed Reality Systems: User Permission Differentiation in Augmented Alma

Supervisor:
Prof. Lorenzo Donatiello

Candidate:
Andrea Schinoppi

Co-Supervisor:
Prof. Gustavo Marfia

First Session

Academic Year 2023/2024

Abstract

The world of extended reality is growing rapidly, and this opens up numerous possibilities for developers.
However, just as with traditional application development, security and privacy issues should not be overlooked, even considering the specific and unique risks of devices that enable extended reality.
In particular, this thesis will focus on improving and strengthening these aspects within the Augmented Alma collaborative platform, developed at the Virtual and Augmented Reality Lab at the University of Bologna.
Features and limitations of this software will be analyzed and then the process by which security and privacy have been strengthened, especially through a new user differentiation system, will be described.
Being able to differentiate users through a level of privilege linked to capabilities has proven to be crucial in protecting the privacy and secrecy of communications within collaborative sessions.

Sommario

Il mondo della realtà estesa sta crescendo rapidamente e questo apre a numerose possibilità per gli sviluppatori.

Tuttavia, proprio come per lo sviluppo di applicazioni tradizionali, i temi di sicurezza e privacy non dovrebbero essere trascurati, anche considerando i rischi e gli attacchi peculiari a cui sono soggetti i dispositivi che permettono di vivere la realtà estesa.

In particolare, questa tesi si concentrerà sul miglioramento e sul rafforzamento di questi aspetti all'interno della piattaforma collaborativa Augmented Alma, sviluppata presso il Laboratorio di Realtà Virtuale e Aumentata dell'Università di Bologna.

Verranno analizzate le caratteristiche e i limiti di questo software e poi verrà descritto il processo con cui sono state rafforzate la sicurezza e la privacy, in particolare attraverso un nuovo sistema di differenziazione degli utenti.

La possibilità di differenziare gli utenti attraverso un livello di privilegio legato a ciò che l'utente può effettivamente condividere o percepire si è rivelata fondamentale per proteggere la privacy e la segretezza delle comunicazioni all'interno delle sessioni collaborative.

Contents

List of Figures

List of Tables

Chapter 1

Privacy and Security in Extended Reality

According to Giaretta (2022), under the umbrella term of Extended Reality (XR), it is possible to distinguish **two** main paradigms: Virtual Reality (**VR**) and Augmented Reality (**AR**).
Virtual reality refers to a **computer-generated** representation of a three-dimensional environment where physical interactions are allowed through the use of wearable devices with sensors, such as head-mounted displays (**HMD**), or hand gestures (this can be also be referred as "**telepresence**", as stated by Kürtünlüoğlu, Akdik, and Karaarslan (2022)).
In augmented reality digital objects and information are **added** to a real environment and can be seen and interacted with through headsets or other devices with cameras, such as smartphones. According to De Guzman, Thilakarathna, and Seneviratne (2019), the contrary is also possible: real objects can be integrated in virtual environments, this is called Augmented Virtuality (**AV**).
A third paradigm which blends both VR and AR is Mixed Reality (**MR**), where interactions do not happen only in physical or digital world but in **both**.
Figure 1.1 shows where each technology finds a place in the "*Reality and Virtuality Continuum*" introduced by Milgram and Kishino (1994):

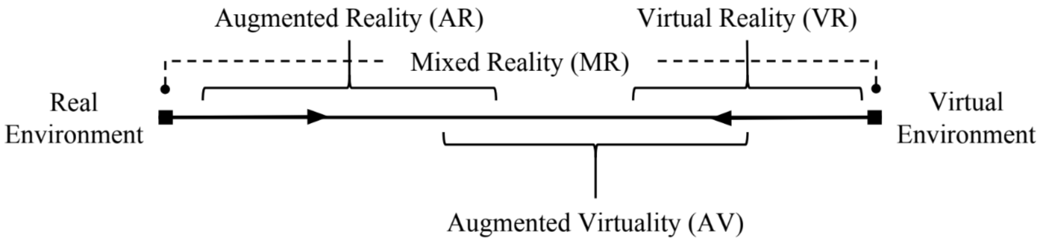


Figure 1.1: Milgram and Kishino (1994)'s continuum (from De Guzman et al. (2019)

1.1 XR related security and privacy concerns

Extended reality devices usually have numerous **sensors** and **cameras**, so they are capable by design of gathering **large volumes** of data both from the **users** and the surrounding **environment**.
With the consideration that extended reality is becoming a very widely-used technology, with an expected global economic **growth** of 287% from 2020 to 2027 (Giaretta (2022)), it is imperative to consider security and privacy aspects of these appliances.

1.1.1 Security and privacy properties

To have a better understanding of the concepts of security and privacy in XR environments, tables 1.1 and 1.2 describe the **properties** and the **guidelines** for both fields that shall be taken into consideration when designing an extended reality system.

Property	**Description**
Integrity	Data shall not be tampered.
Non-repudiation	Data modifications shall be ascribable to the respective author.
Availability	The system shall be always available to accomplish services.
Authorization	Actions should be performed only by authorized users.
Authentication	Only legitimate users shall access the device.
Identification	Actions shall be identified to their actor, unidentified parties can be treated as adversaries.

Table 1.1: Security properties (from De Guzman et al. (2019))

Property	**Description**
Anonymity	Users shall be able to remove their association with data.
Unlinkability	Links with data shall not be distinguishable by adversaries.
Unobservability	Entities' existence can not be ensured or distinguished by an attacker.
Plausible deniability	Users shall be able to deny their relationship with data (especially in case of sensitive data)
Content awareness	Users shall know data, processes and flows which are divulged.
Policy compliance	Systems should follow policies that aim to protect user's privacy or security.

Table 1.2: Privacy properties (from De Guzman et al. (2019))

An additional property which is shared by both privacy and security is **Confidentiality**: actions involving sensitive data shall follow authorization and access control policies.

1.2 Specific XR threats

Being digital devices, HMDs suffer from **general-purpose attacks** such as denial of service or man-in-the-middle, however, extended reality systems are prone to very **specific attacks** related to their particular characteristics, so this section will outline some of the major threats and possible solutions which can be found in the literature.

Figure 1.2 describes a mixed reality environment and resumes how the data generated by an headset could be elaborated by third party services.

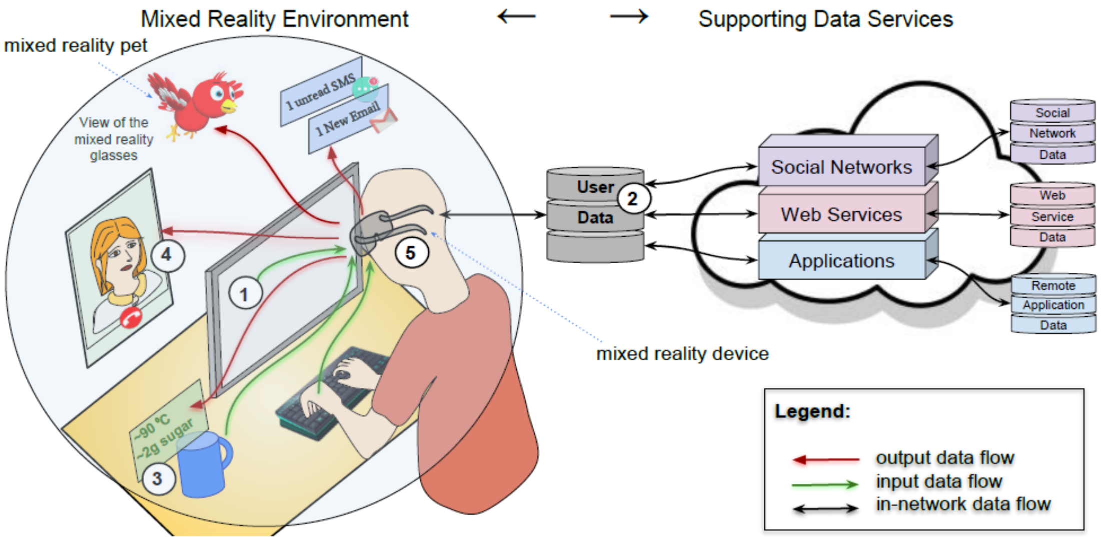


Figure 1.2: Virtual environment and data elaboration (from De Guzman et al. (2019))

In this figure it is possible to see the first five **weak points** concerning security and privacy, which will be explored below, based off work presented by De Guzman et al. (2019).

1.2.1 Input threats

The inputs of the HMD which are generated by sensors and cameras may contain **sensitive data**, both of the user and of other people sharing the same space (the

so called **bystanders**).
For example, a user might use an headset while working on their workstation, allowing the cameras to capture sensitive data on the desktop or colleagues in the same room.

This problem may be solved by adding an **intermediate layer** of protection which applies input **sanitization** techniques.
These techniques may be implemented to act automatically or to follow user defined policies and their aim is to **hide or alter** sensitive information, such as faces or notes, before allowing other third party applications to have access to the input data.
For example, faces and texts are often **blurred** and limbs are converted into a minimal digital representation of the necessary articulations to allow the application to work while preserving the least privilege principle[1].

1.2.2 Data threats

After data has been generated by XR devices, the process of **collection** by third party applications may cause users to **lose control** of their information, for this reason it is important to protect users' privacy during this phase.

Some of the possible solutions involve **pseudonymization** of data (which is not applicable to single users, but to groups only) or the addition of another intermediate layer which allows applications to access only certain sensors data.
Other methods require **encryption** or **secret sharing** techniques to ensure that sensitive information can not be processed by untrusted parties.
Another solution would be providing the applications a "**3D salted reconstruction**" of the surroundings which hides unnecessary details but allows the software to work correctly.
This is slightly different from what was explained in 1.2.1 since it does not refer to the data directly received by the application itself from the sensors but to the data which the application is able to send to its servers to be processed.

1.2.3 Output threats

Once data are processed, the XR system renders a coherent output for the user, which may be **accessed and modified** by untrusted applications or attackers who are physically in the same space (shoulder-surfing-attack). For this reason it is necessary to ensure a private and reliable output.

[1] https://en.wikipedia.org/wiki/Principle_of_least_privilege

In order to prevent shoulder-surfing, **content hiding** techniques can be applied. One example is exploiting different frame rates to allow only the user to see the display content.
Furthermore, a method to hide information from third party applications is **visual encryption** which can be implemented through visual ciphers which allow only legitimate users to decrypt the output.

1.2.4 User interactions threats

During interactions in shared XR spaces attackers may carry out **malicious actions** towards other users or personal information may be leaked to third parties without consent. An example may be a team of competitive XR video games players who need a private space to discuss strategies within a shared session in which their adversaries take part.
Some examples of shared state attacks are documented by Slocum, Zhang, Shayegani, et al. (2023) and involve obtaining users' private information or tampering with what other users can see through various techniques.

A viable solution for shared and private spaces is to **physically divide** them, as mentioned in the previous example, in order to allow users to have their own confidential space.
It is also possible to implement a **soft-concept** of privacy, allowing users to have access to others' private spaces sending a notification both to them and the owner.

The problem of shared spaces is not limited to the spaces themselves but also to the **information** exchanged by users and the digital objects present.
Giunchi, Bovo, Numan, and Steed (2024), for example, have created a framework called StreamSpace through which users can share their screen in shared sessions, being able to **dynamically choose** whether the screen can be visible or blurred so that it cannot be seen by other participants.
An example of multi-screen streaming in a collaborative environment is presented in figure 1.3.

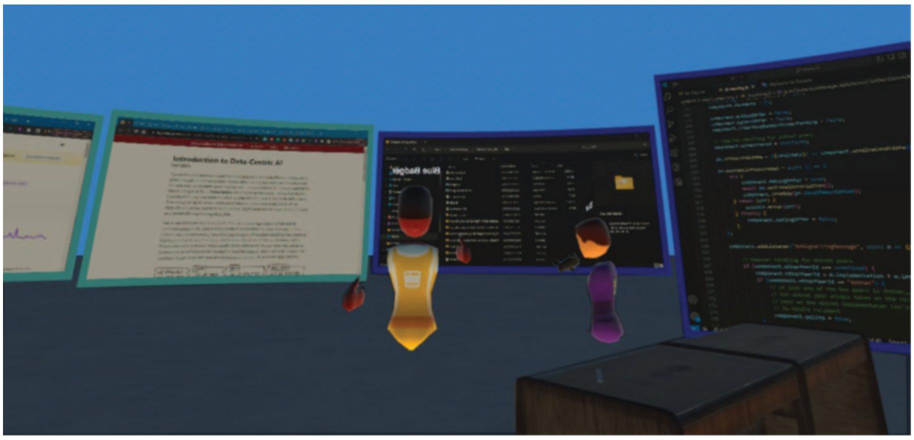


Figure 1.3: Multi-screen streaming in a collaborative environment (from Giunchi et al. (2024))

Regarding personal digital objects or information concealing, two solutions, especially in MR, may be "**privacy lamps**" and "**vampire mirrors**" which users can use to mark items as private and hide them from the others as introduced by Butz, Beshers, and Feiner (1998) and shown in figure 1.4.

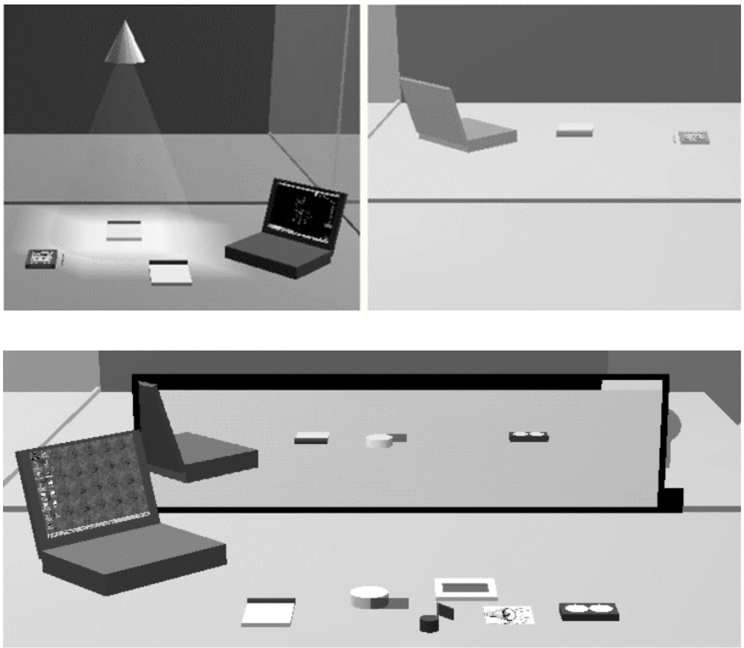


Figure 1.4: Privacy lamp (*Above*) and Vampire mirror (*Below*) (from Butz et al. (1998))

1.2.5 Device level threats

Physical access to XR devices allows hackers to carry **device-level attacks**, for this reason it is important to protect the device itself in order to protect also the data which flow trough it.

The most applied techniques are the so-called **optical strategies** in which the content of the display is hidden by polarized filters or camouflaging techniques, as explained by Pearson et al. (2017).
Another method is **visual cryptography**, as illustrated in subsection 1.2.3.

1.2.6 User identification through HMD data

As mentioned earlier, the presence of numerous sensors in headsets allows malicious applications to **derive several information** from users.
In particular, it has been shown by Slocum, Zhang, Abu-Ghazaleh, and Chen (2023) that, through the processing of gyroscope data and a machine learning model, an attacker is able to perform **keylogging** to find out what the user is typing on a virtual keyboard at any given time.

In the context of authentication, this puts traditional methods such as PIN or password at risk.
However, one possible solution to counteract keylogging are **3D patterns**, in which the user is required to reproduce the correct sequence of elements on the screen with a third dimension added, as shown in figure 1.5

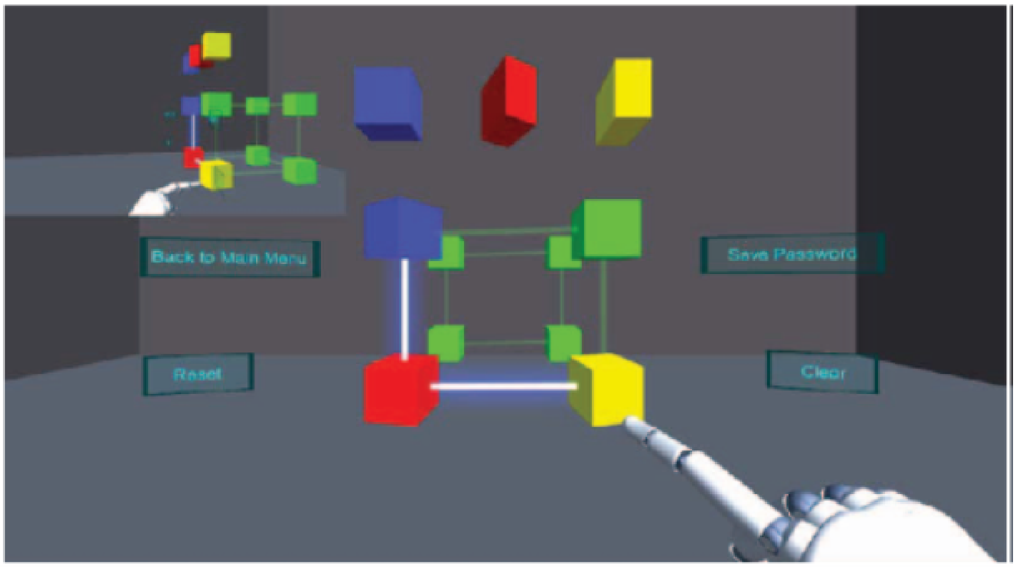


Figure 1.5: Authentication through 3D patterns (from Kürtünlüoğlu et al. (2022))

Nair et al. (2024) have also been able to derive **33 private personal attributes** such as height, weight, and gender with high statistical significance just by analyzing motion data obtained from VR viewers (the so-called **motion tracking "telemetry" data**).

One method of defending against these attacks would be to remove or **restrict sensor access** to applications, however, this could lead to **malfunctions** in background applications even before threats can be mitigated.
Another defense against these attacks is proposed by Sun, Wang, Xue, and Chen (2024) with PPVR: an approach that involves applying **differential privacy** algorithms to motion data to prevent malicious applications from inferring users' private information which may be used to re-identify them.
A similarly oriented approach is **deep motion masking**, introduced by Nair, Guo, O'Brien, Rosenberg, and Song (2023) which uses deep learning to anonymize motion tracking "telemetry" data.

In addition to sensor data from the headset, users can also be identified by **biometric data**, especially eye movements, as proved by Kasprowski and Ober (2004). On the one hand, this allows for an authentication system that is easier to use and more secure than simple login credentials (Lohr and Komogortsev (2022)). However, on the other hand, because of the very high accuracy of the sensors, the possibility of using this data to identify users with little margin for error even during normal use of the headset arises.

Trying to mitigate this risk, Wilson et al. (2024) have proposed a framework to implement a "**Privacy-Preserving Gaze Data Streaming**" that does not compromise the user experience by processing eye data with Gaussian noise, down-sampling, and smoothing techniques.
The pipeline of this framework is shown in figure 1.6.

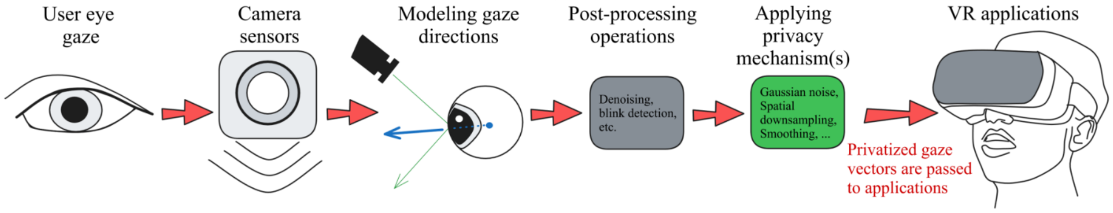

User eye gaze
Camera sensors
Modeling gaze directions
Post-processing operations
Denoising, blink detection, etc.
Applying privacy mechanism(s)
Gaussian noise, Spatial downsampling, Smoothing, ...
VR applications
Privatized gaze vectors are passed to applications

Figure 1.6: Privacy-Preserving Gaze Data Streaming pipeline (from Wilson et al. (2024))

1.2.7 Other threats and attacks

Kürtünlüoğlu et al. (2022) documented **other** threats to XR systems, in particular:

1. **Immersive attacks** i.e. VR specific attacks that exploit immersive characteristic of these devices.
   These attacks are: Chaperone attack, disorientation attack, human joystick attack and overlay attack and they aim at **disturbing the user** by manipulating the environment or the virtual avatar.
   These attacks are strongly related to **motion sickness** as they often disorient the user.

2. **Camera stream and tracking exfiltration** attack, where the attacker becomes able to see through the headset cameras.

3. **Man-In-The-Room** attack, where attackers access the same physical space of the user remaining undetected and infer information from observing the target.

4. **Side-channel** attacks, which exploit secondary channels, such as memory or cache accesses of a certain application to infer sensitive information (also documented by Zhang, Slocum, Chen, and Abu-Ghazaleh (2023)).

5. In some cases XR **interactions** may create unpleasant situations, especially for underage people who may suffer from bullying in the so called Metaverse.
   In extreme cases attacks in XR environments may lead to psychological problems such as post traumatic stress disorder.

1.3 Privacy and security during meetings in XR

Among the many possibilities offered by extended reality are **meetings among people** in the virtual world.

In this setting, the levels of security and privacy required can **vary** widely.
Just as in reality there are different levels of secrecy of a meeting - think of the differences between an informal gathering in a town square, a conference, and a business meeting where trade secrets might be discussed - in the **same way** these events can be **replicated** in the virtual world with the help of HMDs.

It is therefore crucial to protect where necessary the **security and secrecy of communications** from intruders, especially in the virtual world where much more information, such as three-dimensional models, can be shared.

Finally, one last element to keep in mind is user literacy in the area of security:
the **human factor** remains crucial when it comes to cybersecurity.

1.3.1 User differentiation in XR meetings

The peculiarity of meetings in the virtual world is that they take place in a digital space where a mechanism is often implemented to allow people to see and hear events only if they are **sufficiently close** to the source, in order to mimic reality.

It is possible to understand that the only senses involved during these meetings are **sight and hearing**, via the device.
The inability to perceive what is near the user in the absence of external signaling mechanisms puts the privacy of conversations at **risk**.

Consider, for example, two subjects A and B discussing confidential topics. A third person T might get close enough to hear the conversation and perhaps even see elements that should remain secret.
In all this, A and B are unaware of T's presence as they are **out of their field of vision**, while T is able to exfiltrate confidential information undisturbed.

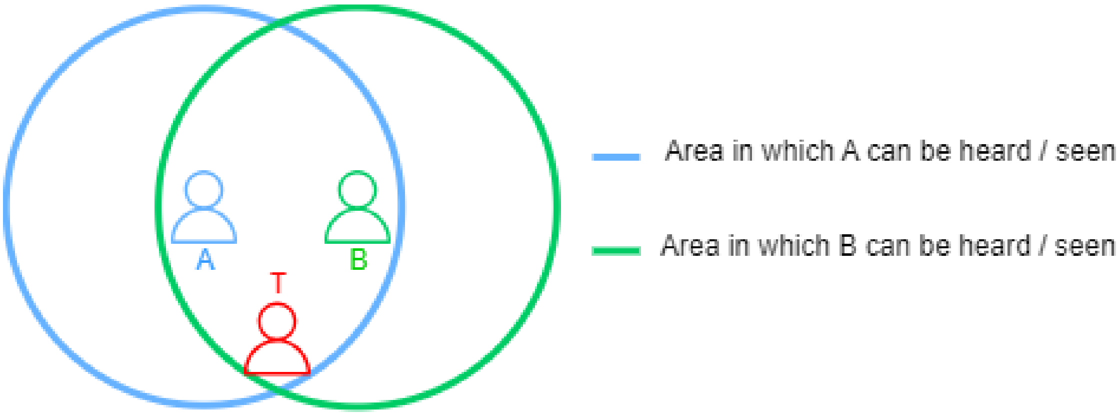

A
B
T
Area in which A can be heard / seen
Area in which B can be heard / seen

Figure 1.7: Depiction of the example

Implementing an **external system** that allows users to notice if other users are outside their perception (for example: a notification system) might be a possible solution, however, it might be desirable to include some users **up to a certain point** in the meeting and later continue with only some others or even exclude some users altogether.

In addition, continuous monitoring of who is in the user's surroundings could considerably **lower** the quality of the meeting.

A possible solution could be a system of **rooms** accessible only by users with certain permissions.
These permissions should be **dynamic**, so as to include or exclude both individual users and categories of users, while the rooms should be inaccessible to

unauthorized users and the information that is exchanged within them should not be available outside.
This would ensure the secrecy of communications and any attachments shown in the context of the meeting.

However, it may be desirable to keep multiple categories of users within the same room but limit only the **interactions** they can have.

A possible solution, according to Marques, Silva, and Santos (2023), involves applying policies to **negotiate** access on contents with other users.

Consider, for example, a presentation in XR in which the audience is allowed to speak only at the end: in order to prevent someone from taking the floor at an inappropriate time, the host could inhibit the audience from using voice interactions.
Or think of a meeting with a high level of secrecy in which at some point a user wishes to share their screen, a three-dimensional model, or other confidential information only, for example, with other executives: in this case, even lower-ranking figures could be allowed to remain in the room, but without the possibility of seeing the confidential information.

Clearly, these permissions should also be **dynamic** so that users' ability to interact can be re-enabled at any time.

Moreover, dynamic permissions could also help users to decide **which** kind of data to share with third parties, which, at the moment, is an open issue in the field of privacy in XR collaborative applications (Marques et al. (2023)).

Chapter 2

Augmented Alma

This chapter will introduce the XR collaborative platform **Augmented Alma** on which this thesis project was carried out.

The main features and their limitations in terms of privacy and security will be explained, and the proposed resolution to these problems will be presented in Chapter 3.

Before introducing Augmented Alma, it is important to note that the platform is still in a **state of development**, and up to this point the issues of privacy and security had not yet been addressed in favor of features that would bring value to the end user, therefore work was required to start from the foundations and then work up to the individual features.

2.1 What is Augmented Alma?

Augmented Alma is a collaborative virtual reality system developed at the virtual and augmented reality laboratory of the University of Bologna (VARLab[1]).
It is based on the engine **WiXaRd**, presented by Stacchio et al. (2024), which in turn takes advantage of the **ATON** framework[2] which provides a solid base for collaborative experiences and its front-end Hathor, both created by Fanini, Ferdani, Demetrescu, Berto, and d'Annibale (2021).

ATON is an open source framework based on Node.js and Three.js that allows creating and manipulating scenes within responsive WebXR applications without requiring additional installations by end users.

[1] VARLab website: https://site.unibo.it/varlab/en

[2] ATON website: https://osiris.itabc.cnr.it/aton/

It also natively supports the creation of virtual tours with annotations in the scenes (even in audio format) and three-dimensional models.
Another very important feature of this framework is the possibility to create multi-user sessions.

2.1.1 Main features

Augmented Alma was born as a collaborative platform for **teaching** in which a teacher (even at the university level) interfaces with students by means of numerous tools.

More specifically, the lecturer can interact with a **3D scene** that students can view, possibly interacting in turn.

It is clear this platform was created for teaching purposes but its functions can also be used during **general purpose** meetings or even business meetings and not only by students and professors.

Thanks to WiXaRd, the platform supports **remote rendering**, which allows a 3D scene or object to be loaded and streamed to both the teacher's device and those in use to the students, which could be not only VR/AR viewers but also smartphones and PCs.
Furthermore, the platform allows two-way communication, both video and audio, and through spatial **annotations** that users can enter and can be seen by others.

What has just been illustrated is the "**collaborative mode**", however, the system also offers a "**remote streaming mode**" in which only the main user (e.g., teacher) has full control of the interaction with the scene: other users (e.g., students) will be able to connect to a video stream of what the main user sees, still with the possibility of entering and reading any annotations on the scene.

Figure 2.1 shows the architecture of Augmented Alma project.

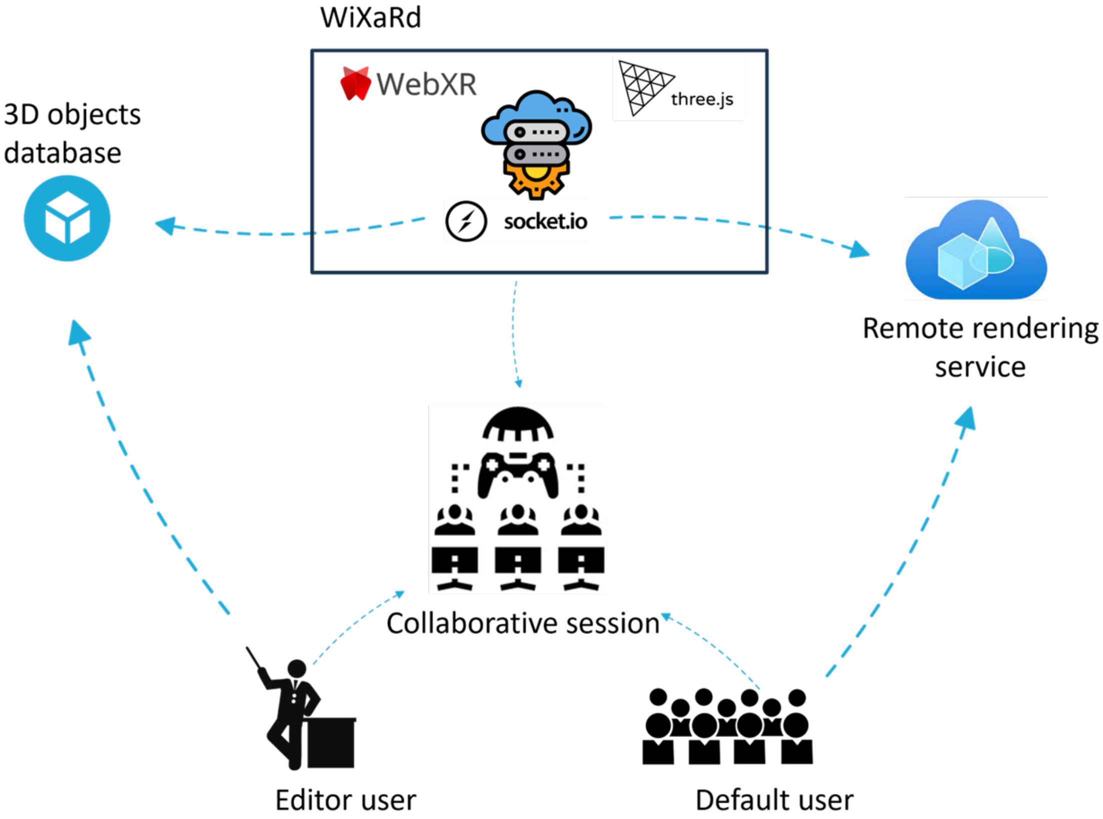


Figure 2.1: Augmented Alma architecture (from Stacchio et al. (2024))

Annotations

There are three types of annotations:

1. **Simple annotation**: a flashing sphere will appear indicating where the annotation was inserted;

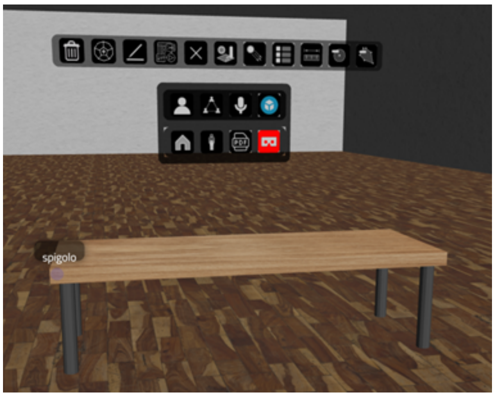


Figure 2.2: Example of simple annotation

2. **Geometric annotation**: with which user can highlight a user-defined area where where to insert the annotation;

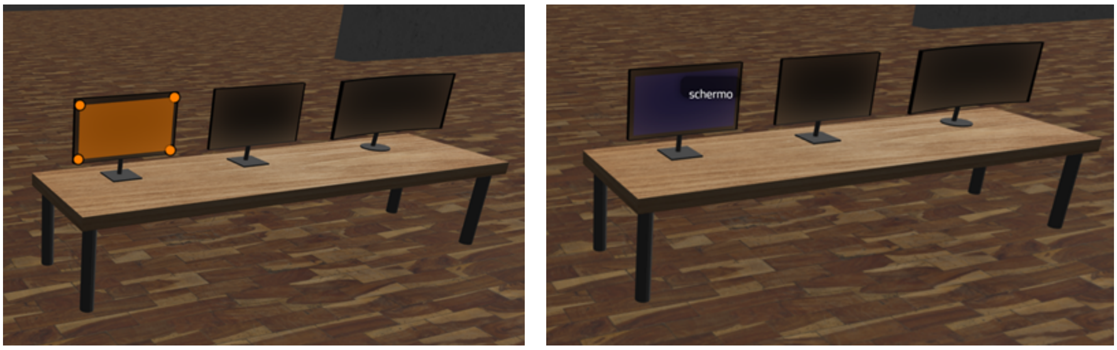


Figure 2.3: Example of geometric annotation

3. **Freehand annotation**: free drawing on objects in the scene.

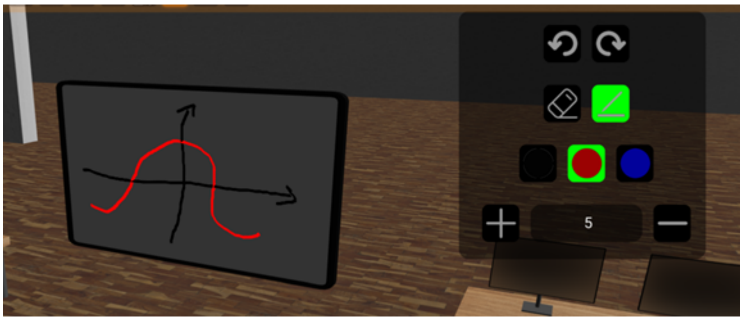


Figure 2.4: Example of freehand annotation

On a browser, annotations can also be consulted via an ad-hoc pop-up.

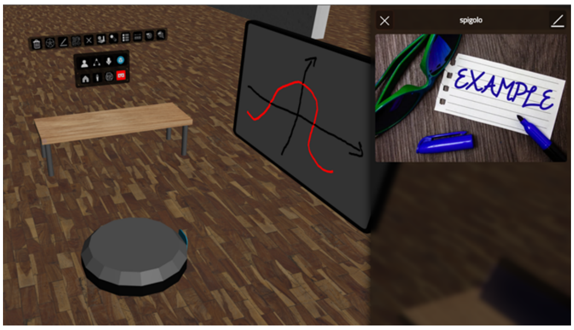


Figure 2.5: Annotations pop-up

3D models

The platform also allows to import 3D objects from a platform called **Sketchfab**[3], which contains several models, but also allows uploads and distribution of 3D assets.
Moreover, Sketchfab guarantees data protection using encryption and provides APIs to be used with a token upon registration.

The model to import can be found via text search or in the user's personal **collection**, which can be created in order to access models in a easier and faster way.

[3] Sketchfab website: https://sketchfab.com/

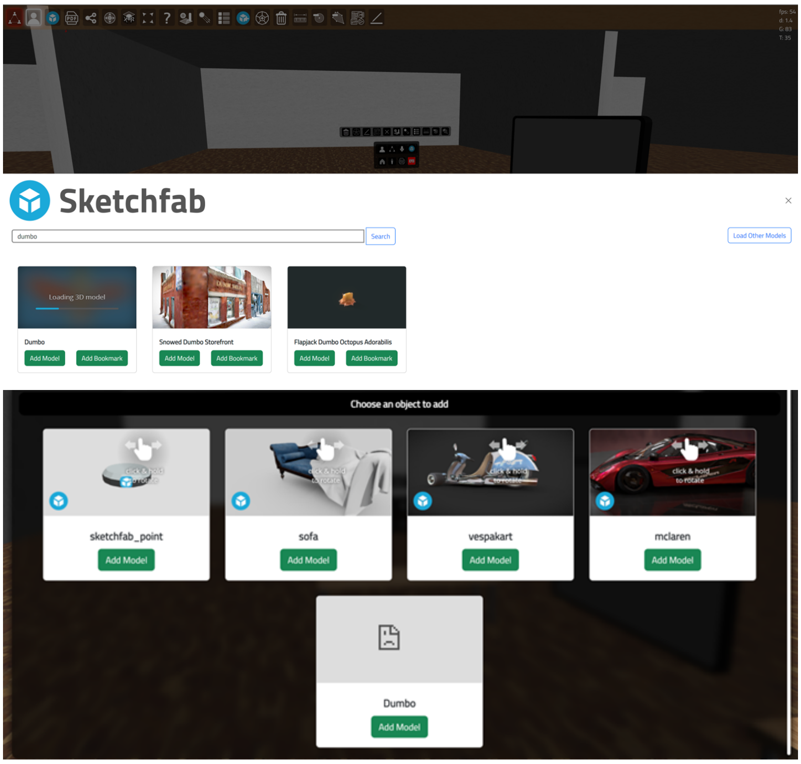


Figure 2.6: 3D models import via text search (*Above*) or personal collection (*Below*)

Manipulation, removal and hierarchy of models are also allowed:

Figure 2.7: Example of 3D models hierarchy

Support for spherical images and 3D models

Augmented Alma allows to decide the appearance of room interiors in two ways:

1. From **360-degree images**, which are applied to a sphere of appropriate size so as to create a space large enough for users.
   An example is the "Entrance" room, represented by Figure 2.8:

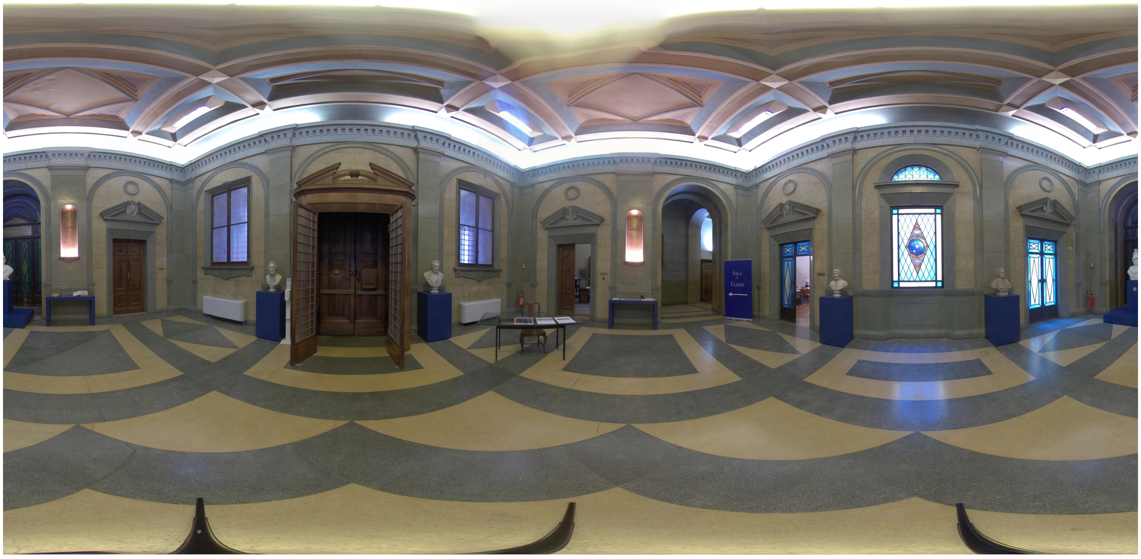


Figure 2.8: Entrance 360 degrees image

2. By importing **3D models**, which are then rendered to allow users to move around inside.
   An example is the St. Cristina complex of which a 3D model is available in Figure 2.9:

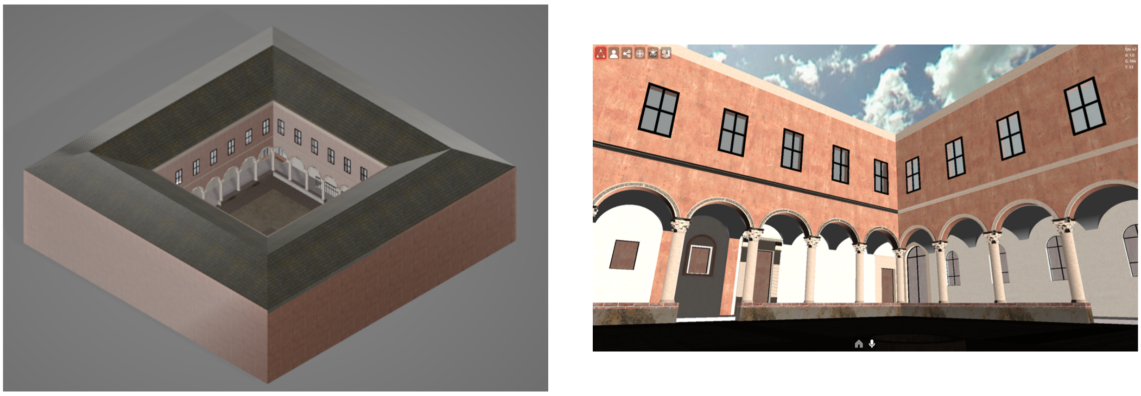


Figure 2.9: St. Cristina complex 3D model (*Left*) and rendering (*Right*)

Virtual tour

Users can visit **different rooms** via the virtual tour interface by simply selecting the one they wish to go to.
This triggers the scene change, showing the selected room.

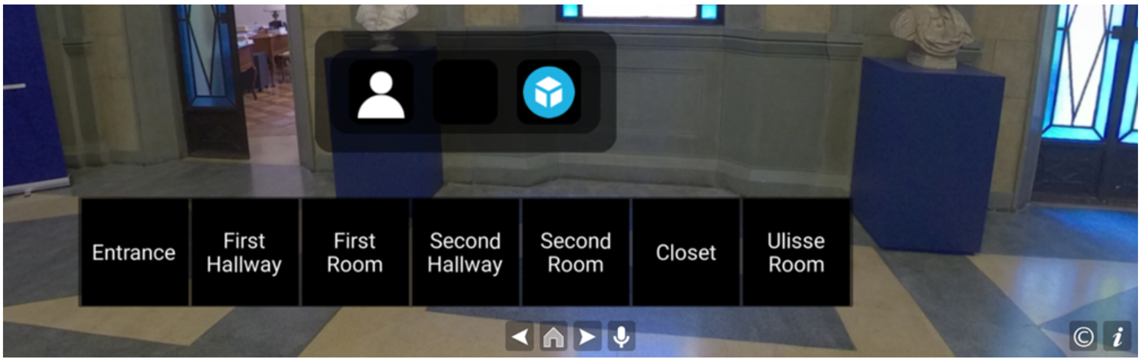


Figure 2.10: Virtual tour interface

Unfortunately, this feature introduces the **first** privacy problem.
In fact, the room change does not correspond to a multi-user session change, causing two users virtually in different rooms to **interact** as if they were in the same room.

Figure 2.11 shows this case: "User #1" and user "ra" are virtually in two different rooms (Entrance and Ulisse Room respectively) but can see each other's movements:

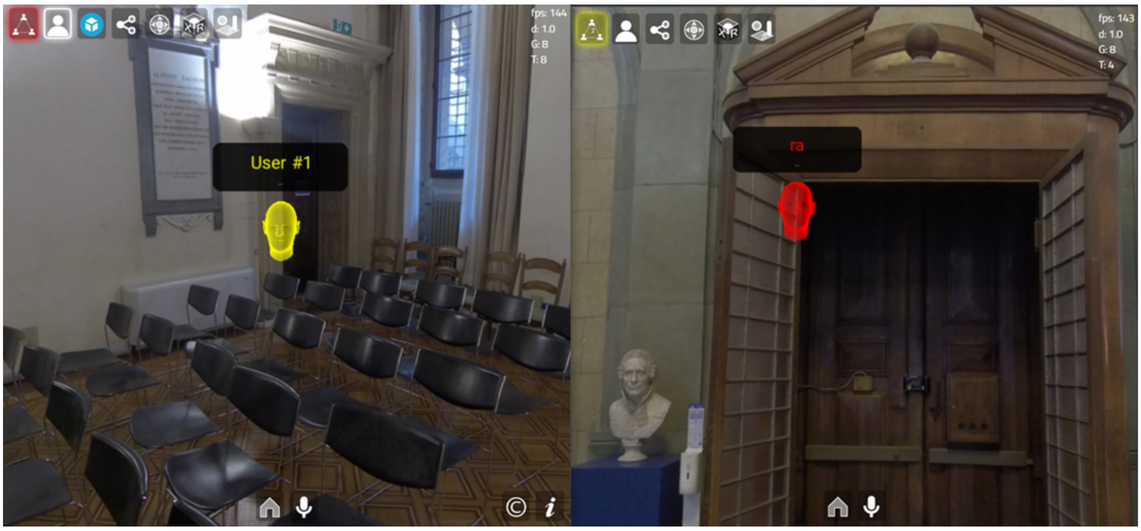


Figure 2.11: Users in different rooms interact as if they were in the same one

Multimedia streaming

During collaborative sessions, users can talk to each other by audio **streaming** and share their webcam or screen with other users.
These are key features within a platform such as Augmented Alma given its educational purpose and, more generally, the applications it might have as a collaborative extended reality app.

However, what is shown in Figure 2.11 unfortunately remains true even in the case of multimedia streaming.
In fact, users within the same session who are in different rooms continue to hear other users and see the associated camera or screen streams.

Figure 2.12 shows user “ra” sharing their screen from Ulisse room while “User #1” is able to see it from Entrance.

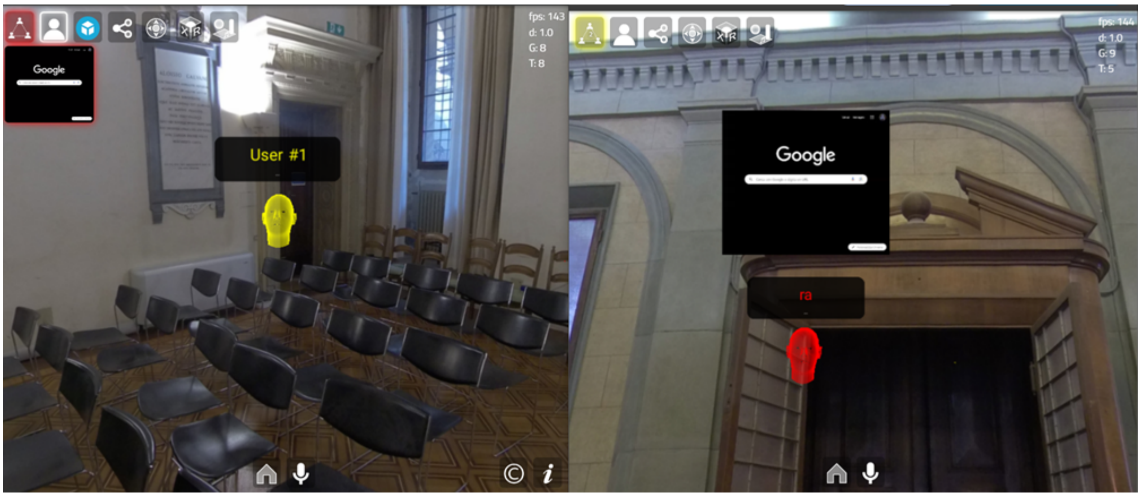


Figure 2.12: User in different room can hear and see other user’s multimedia streams

Considering what was introduced in Section 1.3.1, more specifically in the example in Figure 1.7, the fact that each user can interact with others as if they were in the same room endangers the privacy and secrecy of communications between users since **anyone** is able to hear and see everything without distinction.

User differentiation

The only form of **differentiation** between users implemented is the difference between an administrator and a standard user.

Moreover. there is no possibility of preventing a user from creating or receiving media streams, which was one of the requirements advocated in Section 1.3.1.

Also, since **annotations** might contain sensitive information it is necessary to place access rights on them as well, so that only authorized users can read them. Moreover, a similar argument applies to **3D models**, which in the context of a high-secrecy meeting might be desirable to show only to users with special permissions.

Finally, as Figure 2.10 shows, each user has **unlimited access** to each room, which is undesirable if it is required to have rooms with higher access privileges to protect privacy and communication secrecy.

2.2 Comparison with other platforms

In this section Augmented Alma will be compared with other XR collaborative platforms.
In particular Table 2.1, which was taken from the paper Stacchio et al. (2024), shows the comparison between WiXaRd, i.e., the engine on which Augmented Alma is based, and other XR collaboration platforms in relation to certain aspects. These aspects are:

- XR Device Agnostic (**XRAG**);
- Collaborative (**C**);
- 2D device compatibility (**2DC**);
- Scene Customization (**SC**);
- Remote Rendering (**RR**);
- Presence of an Annotation System (**ANN**);
- External Database Integration (**EDI**);
- General Purpose (**GP**);
- Open-source (**OS**)

The platforms will be divided between industrial and academic ones.
The **industrial** platforms are:

[1] Varjo Reality Cloud[4]

[2] Meet in VR[5]

[3] TechViz Virtual Reality Cloud[6]

[4] Meta Immersive Learning[7]

[5] Microsoft Mesh[8]

The **academic** platforms are:

[6] A collaborative WebXR platform for medical learning by Al Hafidz et al. (2021);

[4] Varjo website: https://varjo.com/products/realitycloud/
[5] Meet in VR website: https://www.meetinvr.com/
[6] TechViz website: https://www.techviz.net/en/cloud-and-viz/
[7] Meta website: https://about.meta.com/it/immersive-learning/
[8] Microsoft website: https://learn.microsoft.com/en-us/mesh/overview

[7] A web-based collaborative XR app introduced by Korečko et al. (2021);

[8] A WebXR platform for touristic and cultural services by Martí-Testón, Muñoz, Gracia, and Solanes (2023);

[9] A framework for remote collaborative interactions in XR, created by Pereira, Matos, Rodrigues, Nóbrega, and Jacob (2019);

[10] ATON framework by Fanini et al. (2021), introduced in Section 2.1.

	XRAG	**C**	**2DC**	**SC**	**RR**	**ANN**	**EDI**	**GP**	**OS**
	Industrial								
[1]	✗	✓	✗	✗	✓	✗	✗	✗	✗
[2]	✗	✓	✗	✗	✗	✗	✗	✗	✗
[3]	≈	✓	✓	✓	✓	✓	✗	≈	✗
[4]	✗	✓	✗	✓	✗	✗	✗	✓	✗
[5]	✗	✓	✗	✓	✓	✓	✗	≈	✗
	Academic								
[6]	✓	✓	✗	✗	✓	✗	✗	✗	✗
[7]	✗	✓	✓	≈	✗	✗	✗	✗	✗
[8]	✓	✓	✓	✗	✗	✗	✗	✗	✗
[9]	✓	✓	✗	≈	✗	✗	✗	✗	✗
[10]	✓	✓	✓	≈	✗	✓	✗	✗	✓
Augmented Alma	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2.1: Comparison among systems considering relevant features (from Stacchio et al. (2024))

None of the technologies presented explicitly refers to the **differentiation** of users not only from the point of view of the difference between administrator/organizer/co-organizer and ordinary user attending a meeting, but also from the point of view of the capabilities they have within the meeting.

Some, especially of industrial nature, allow higher-level users to mute microphones or block other participants' screen sharing, but do not give the ability to limit what they can see or hear.

For example, Figure 2.13 shows a host panel in Microsoft Mesh ([5]).

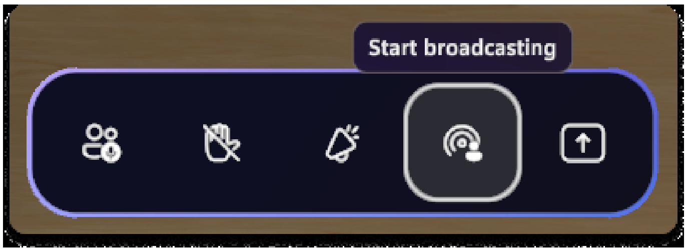


Figure 2.13: MS Mesh host panel

From left to right, the meeting host can turn off the microphone to all users, disable hand raises, turn on the megaphone function to make themselves more audible, start a broadcast (for multi-room events), or share their screen.

The lack of a system for differentiating users and rooms is a problem that impacts evenly **all** of the listed platforms and puts the secrecy of communications at risk. For this reason, all the applications mentioned above would in any case **benefit** from a system of rooms and users permissions differentiation such as the one that will be introduced in Chapter 3.

2.3 Project tree structure

The following figure shows the tree structure of the Augmented Alma project:

```
AugmentedAlma ................................................ Root folder
├── config ...................................... Configuration files and DBs
│   ├── certs ..................................... HTTPS certificates
│   ├── (users.json)
│   └── (rooms.json)
├── data .......................................... ATON data folder
│   ├── [...]
│   ├── logs ...................................... Logs folder
│   │   ├── [...] ......................... One log folder per user
│   │   └── (security_log.csv) ........... General security logs file
│   └── scenes ................................ Published 3D scenes
│       ├── [...]
│       └── samples
├── [...]
├── node_modules
├── public
│   ├── [...] ........................................... Utilities
│   ├── hathor ........................................ Hathor files
│   │   └── (hathor.*.js files)
│   ├── res .............................. Resources (images, icons etc.)
│   │   └── [...]
│   └── src ......................................... ATON sources
│       │   └── (ATON.*.js files)
│       ├── ATON.sui ........................................ UI files
│       └── [...]
├── services ............................. Server, APIs and services
│   ├── (ATON.service.main.js) ........................ Express server
│   ├── (API.js / newAPI.js)
│   ├── photon ...................................... Photon folder
│   └── [...]
├── sessions ....................................... Users' sessions
└── (Proxy.service.js) ................................ Proxy server
```

Figure 2.14: Tree structure of Augmented Alma project

Starting from the top, the `config` folder contains both https certificates and the user and room databases.
The `data` folder contains mainly the data useful to ATON, the scenes that give the appearance to the rooms (`scenes` subfolder), and the log files.
The `public` folder contains the sources for Hathor front-end and ATON, including resources (such as images). These files are minified at compile time in order to make them lighter.
The `services` folder mainly contains the Express.js server with its APIs and the `photon` subfolder where the logic that enables multi-user sessions is located.
Finally, the `sessions` folder stores user sessions, while the `proxy.service.js` file provides a proxy server to which ATON connects.

Chapter 3

Privacy and security improvements

This chapter will outline the improvements implemented to solve the problems presented in Chapter 2 and the new functionalities introduced in the context of this thesis work.

3.1 Authentication API

The first improvement to security was the creation of new APIs for user **authentication**.

Previously, each user in the database was defined by the `username`, `password` (stored in plain text), and "`admin`" fields - the latter being a boolean value to indicate whether or not they were administrators.
An example user is shown in the following JSON snippet:

```
{
    "username": "User",
    "password": "Password",
    "admin": False
}
```

Authentication was performed by comparing the payload of the login request sent by the user with the users in the database using Passport.js.

After the changes, the users' passwords are saved after being **hashed** with salt, so that in case of malicious access to the database, the users' passwords cannot be reconstructed.

Authentication is still performed using Passport.js by verifying via the Bcrypt library whether the password entered by the user during login matches the stored salted hash.

The authentication system was also upgraded to provide each user with a **JWT token** after login.

The JSON Web Token (JWT) is an open standard presented in RFC 7519 by Jones, Bradley, and Sakimura (2015) that defines a schema in JSON format for the exchange of information or authentication.
The generated token consists of a `header` with information about the encryption algorithm used, a `payload` containing the information to be exchanged and other information such as the token's expiration date, and a `signature` that is created using a server-side secret key: this ensures that if the token is corrupted or modified by an external agent, it will not pass validation.
Header and payload are encoded in base64 format and then encrypted with the secret key with the algorithm in the header.

In the case of Augmented Alma, the secret key for signing server-side JWT tokens was generated using the command "`pwgen -s 64 1`"[1] and placed in an `.env` file that was not made public.

After successful login, each user receives a JWT token to save and use for subsequent requests.

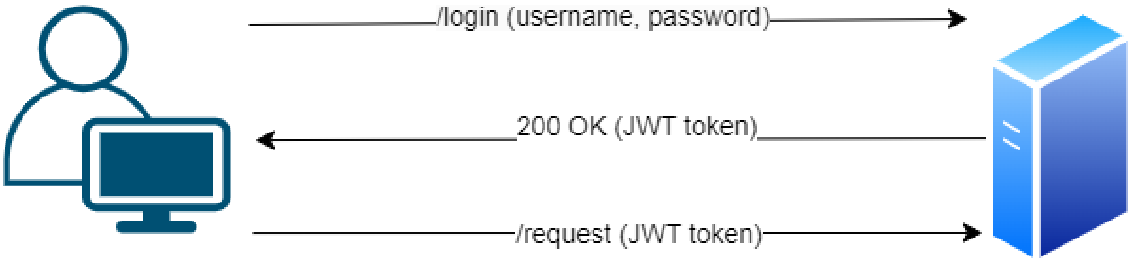


Figure 3.1: Authentication and JWT token

The JWT token **expires** after one hour, but if the user is still online, it is automatically **refreshed**.

If the user logs out, their JWT token is automatically **deleted**.

[1] Pwgen man page: https://linux.die.net/man/1/pwgen

3.2 Rooms and sessions distinction

The next intervention dealt with the problems highlighted in Figures 2.11 and 2.12 concerning the possibility of interaction between users who are virtually in different rooms.

To solve them, it was necessary to create **new sessions** based on the room in which the users are located.

First, in order to switch from a simple scene change to actual separate rooms, it was necessary to create a database that contained all available rooms so that it was possible to assign each room a **unique URL** that would allow users to reach it. Initially, the rooms had the following fields:

```
{
        "name": "entrance",
        "url": "/entrance",
        "sceneID": "samples/entrance"
}
```

Once each room has a URL it is necessary to create **dynamic routes** for the Express.js server so that all rooms can be reached by users easily.
To do so, the database of rooms is parsed and all routes inserted before the last route which is the one that intercepts invalid URLs.

3.3 Users' levels

The third improvement relates to user **privilege levels**.

The boolean value "`admin`" presented in Section 3.1 has been transformed into an integer called "`level`" that ranges from 1 to 5 and indicates the user's privilege level.
An example user is shown in the following JSON snippet:

```
{
    "username": "User",
    "password": "$2b$10$joMCpj2MqNJyVioK1ZM6y0B.k[...]",
    "level": 3
}
```

This change was the **turning point** for user differentiation since it allows five

levels to be distinguished, the highest of which corresponds to what was previously the boolean `admin`.
One of the advantages of this setup is that the system could potentially handle **as many** levels as needed, also way more than five.

This opens up many possibilities, which will be discussed in Sections 3.4 and following.

3.4 Rooms access

Once the rooms were made accessible via URL, work continued by creating a distinction among rooms based on **access permissions**: each room was assigned a minimum privilege level to access it:

```
        "accessPermissions": 3
```

Unlike user levels, this value ranges from 0 to 5, so that **non-authenticated** users (level 0) can access some rooms, including the landing room.

To reflect this change on the user experience, the virtual tour interface has been made **dynamic**, showing only the rooms that the current user can actually access, as shown in Figure 3.2.

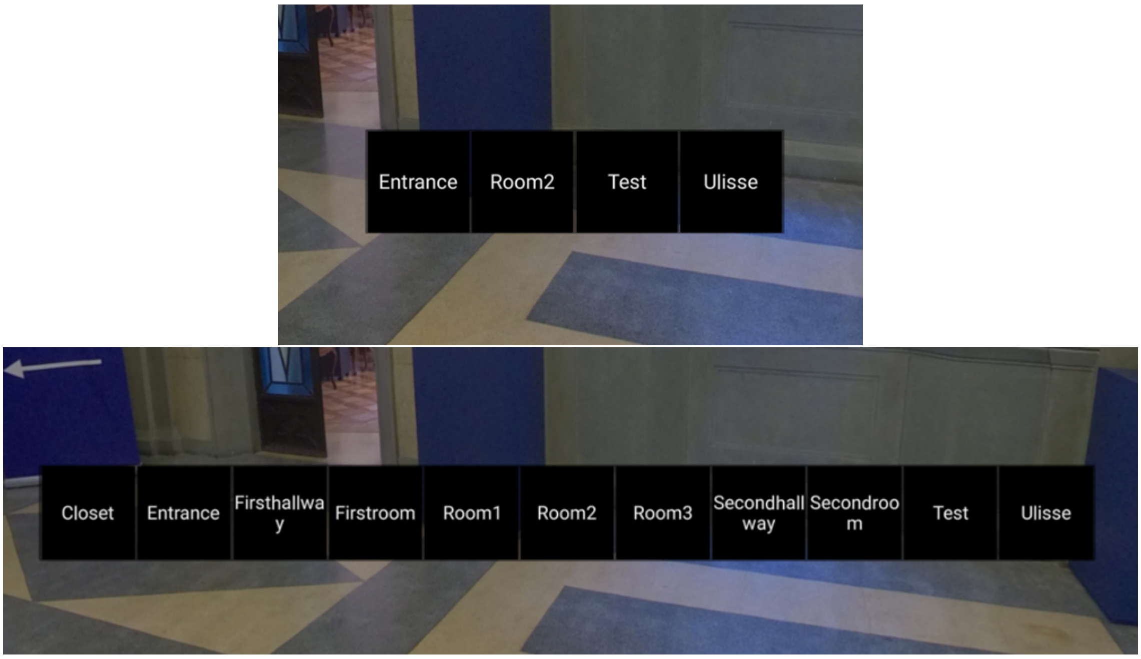


Figure 3.2: Dynamic virtual tour interface with user level 0 (*Above*) and 5 (*Below*)

In the event a user knows the URL of a room they cannot access and enters it in the browser search bar, the access will be **precluded** and they will be notified of this via an alert and redirected to the page they came from.

3.5 Multimedia authorizations

The focus of this thesis work developed from an example similar to the one in Figure 1.7, wondering whether it would be possible to prevent third parties sharing the same virtual space from hearing or seeing the information that users exchange in order to protect the secrecy of communications and the privacy of the users themselves.

Indeed, it is not not to be taken for granted that all third-party users have bad intentions such as exfiltrating private information, but just as in physical reality it is undesirable to be eavesdropped on, similarly this is also true in extended reality.

Especially in collaborative applications such as Augmented Alma, where meetings and gatherings may involve topics of **varying secrecy**, dynamically placing limits on the information users can receive is very important.

Following what was discussed in Section 1.3.1, a system was then implemented that, by taking advantage of user privilege levels (see Section 3.3) manages permissions to perform multimedia streaming.

Such a system requires each room to have stored the **minimum** privileges necessary to stream or receive audio, camera, or screen and to display 3D models and annotations.
The structure of the rooms within the database was then updated by adding the following fields:

```
        "CameraSendPermissions": 0,
        "CameraReceivePermissions": 0,
        "MicrophoneSendPermissions": 1,
        "MicrophoneReceivePermissions": 2,
        "ScreenSharePermissions": 3,
        "ScreenReceivePermissions": 3,
        "modelsReceivePermissions": 4,
        "AnnotationsReceivePermissions": 5
```

Each of these eight new values ranges from 0 to 5 and represents a permission that is compared with the user's level to determine whether or not they can send or receive certain data.

3.5.1 Audio and video streaming

As for the first six, they concern the stream and reception of audio, camera, and screen.
The data propagation system is **event-based**: when the user initiates a stream, the data is sent in packets via events to all users in the same session who proceed to decode and play it back.
In the cases of camera and screen, a small window appears above the user showing the video stream (as can be seen in Figure 2.12), while in the case of audio, the avatar of the person who is speaking will show sound waves around the mouth so that the source of the sound can be visually identified.

In implementing user differentiation, streaming data sending and receiving had to be taken into account separately.
In the case of sending, when the user presses on one of the available buttons to start one of the three streams, it is checked whether the permissions are sufficient, and if not, the user will be **prevented** since the beginning from starting to stream.

In the case of receiving, on the other hand, it has been chosen not to simply hide other people's streams from the user who does not have sufficient permissions.
For example, it would have been possible to not let the user hear the audio or to hide the window showing the videos, but this would not have prevented the data packets from arriving, potentially allowing reconstruction of the information.
It was therefore chosen to **block the packets** from arriving by acting at the origin on the corresponding event handlers: if the user cannot access a certain stream the corresponding packets will not be decoded by the middleware but discarded.

In this way potentially the user does not even realize that a streaming is in progress.

Figure 3.3 shows the point of view of two users: one can watch a screen sharing stream, while the other cannot even see that a stream is in progress.

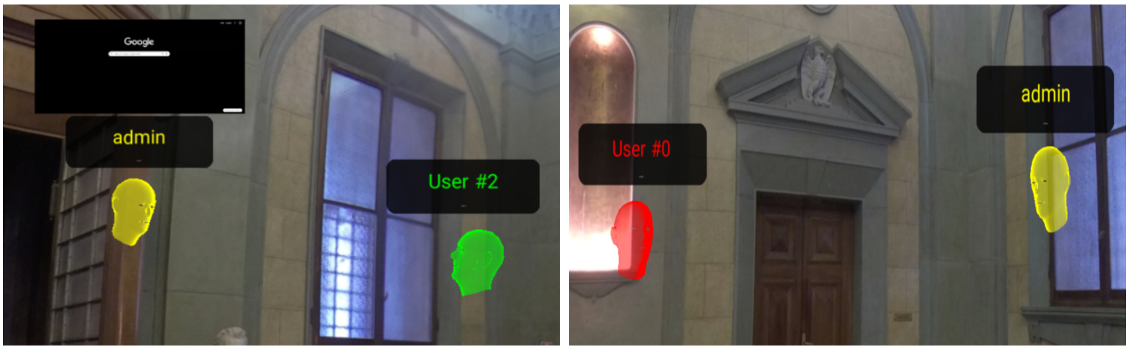


Figure 3.3: "User #0" can see screen sharing streams (*Left*), while "User #2" can not (*Right*)

3.5.2 3D models and annotations

A different argument needs to be made regarding 3D models and annotations. As anticipated, there are several reasons why it might be desirable to hide annotations and 3D models, and the main ones concern the information they might contain or represent.

Similar to what is defined by the WiXaRd specification (Stacchio et al. (2024)), each scene is defined by a `scene.json` file that contains all the information needed for the render, such as the position of the lights and the surrounding environment.

Along with this information, a scene graph and a semantic graph are also stored: the former contains the 3D model of the room, which can be a sphere to which an image is applied or an actual 3D file, and information about any imported 3D models that may be present.
The second, on the other hand, contains all annotations information, including textual and audio descriptions.

Since both annotations and 3D objects are part of the scenes themselves, the approach taken to prevent users with insufficient permissions from accessing the information is to **not render** the semantic graph at all and restrict the scene graph rendering only to the room itself.
An example of this implementation for 3D models is shown in Figure 3.4.

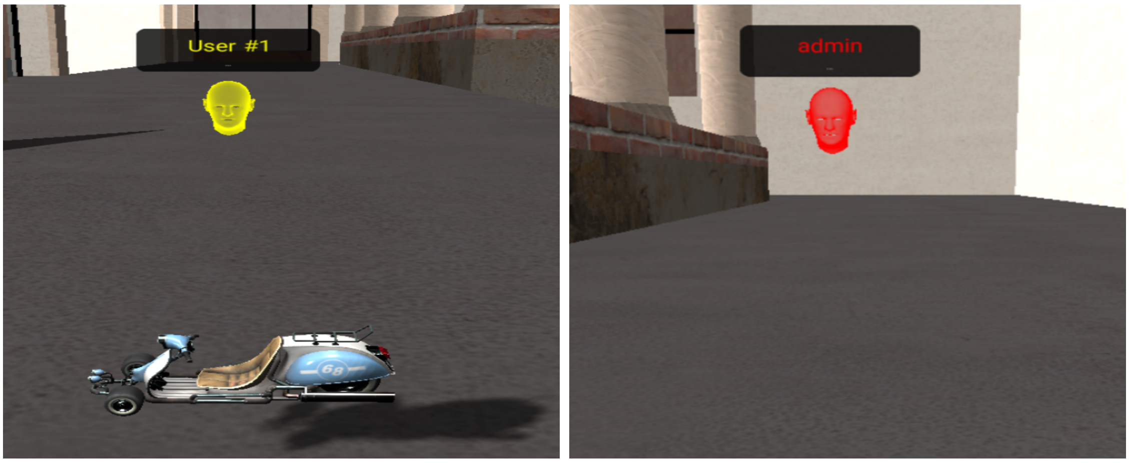
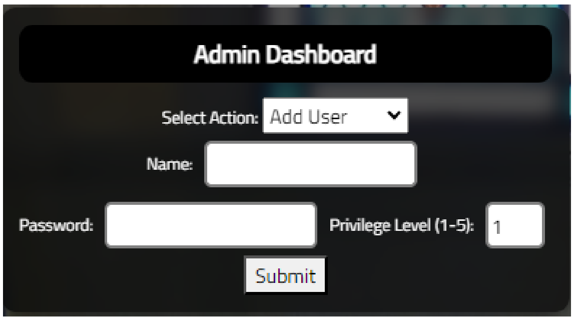


Figure 3.4: User "admin" can see 3D models (*Left*), while "User #1" can not (*Right*)

3.6 Administrator dashboard

In order to meet the requirement of dynamic permissions, proposed in Section 1.3.1, it is necessary that they can be changed.
Therefore, an **administrator dashboard** is introduced: from this dashboard only administrators (i.e., level 5 users) can choose from a dropdown menu and perform several actions, which will be discussed in the following Subsections.

3.6.1 User addition



Select Action: Add User

Privilege Level (1-5): 1

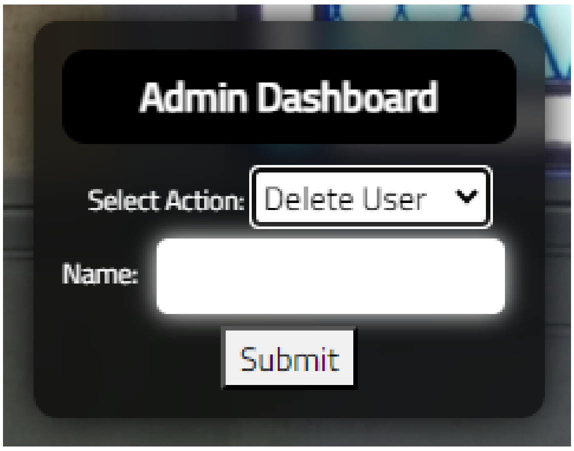
Figure 3.5: Administrator dashboard for user addition

Choosing the first action ("**Add User**") will make possible to add a new user to the database by entering the new username, which must be unique, a password, and choosing the privilege level.

Upon submission, the request, accompanied by user's JWT token, is sent to the server, which adds to the database a user with chosen username, level, and hashed and salted password.

An endpoint for user **registration** has also been implemented but has not yet been equipped with a graphical user interface.

3.6.2 User deletion



Admin Dashboard

Select Action: Delete User

Name:

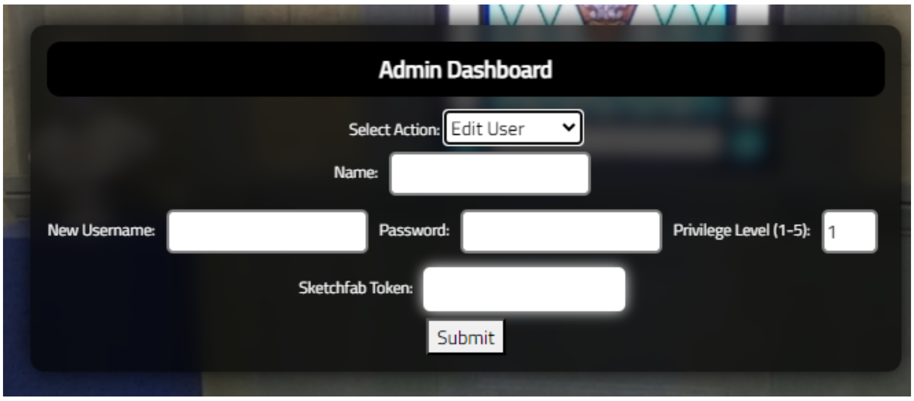
Figure 3.6: Administrator dashboard for user deletion

Choosing the second action ("**Delete User**") will allow the administrator to delete a user: after checking the authenticity of the JWT token, the server removes from the database the user whose username matches (case-sensitive) the one entered in the form.
Since the usernames are unique, there is no risk of deleting other users by mistake.

The operation of deleting a user should take place when the user is not online, however, the case where a user is deleted while on the platform is also handled: in this case the user will be **redirected** to the landing page, automatically logged out, and then notified via an alert of what has happened.
Deleting a user from the database clearly prevents future logins from them.

Administrators cannot delete themselves.

3.6.3 User Edit



Select Action: Edit User

Privilege Level (1-5): 1

Figure 3.7: Administrator dashboard for user editing

Editing a user via the “**Edit User**” option provides the possibility to edit the data of the user whose username matches the one entered in the form.
Specifically, it is possible to change both a user’s username and password, change their privilege level, and add the “Sketchfab token” that allows them to interact with the Sketchfab APIs for 3D models.

The server will edit only the fields that have been filled in the form.

Again, it is recommended that editing of users occurs when they are offline, but the case of editing with an online user is still handled.

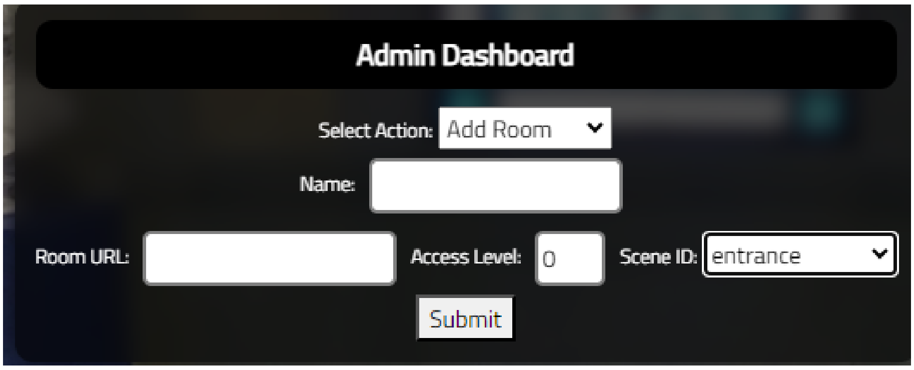
Several circumstances are considered:
If the user name is changed, the user will be **logged out** and redirected to the landing page so that if they wish they can log in again with the new username.
On the other hand, if the privilege level is changed, if it is found to be insufficient to remain in the room the user is in, the user will be **redirected** to the landing page with an alert informing them of the situation.
A new privilege level always triggers a JWT token **refresh** to reflect the modifications.

Depending on changes in their level, a user may no longer be allowed to send media streams - in which case all the affected streams will automatically stop.
As for receiving, on the other hand, if streams are in progress, the user who becomes authorized to receive them will have them available immediately. On the contrary, if the user level becomes insufficient to receive streams, they will dynamically become unavailable.

Finally, 3D models and annotations require a **reload** of the page both when the user loses access and when the user gains access.
This is because it is necessary to re-render the page with or without scene- or semantic- graph.

Administrators cannot edit themselves.

3.6.4 Room addition



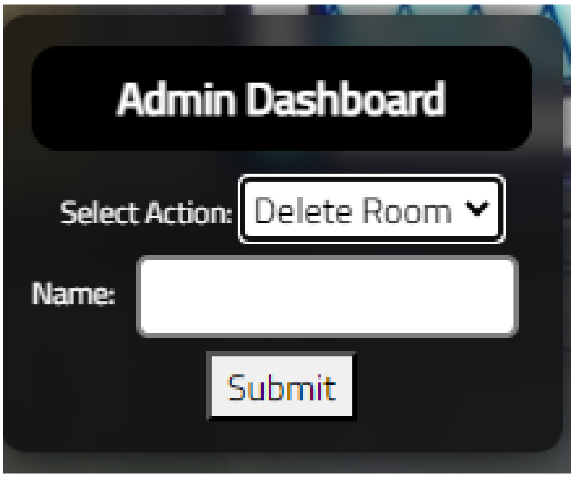
Select Action: Add Room
Name:
Room URL:
Access Level: 0
Scene ID: entrance

Figure 3.8: Administrator dashboard for room addition

With the "**Add Room**" function, an administrator can add a room to the database by choosing its unique name, unique url, access level, and scene ID.
The last parameter represents what the new room will look like and can be chosen from ten alternatives: when a new room is created its scene.json file will be a **clone** of that of the chosen room but without annotations or 3D models.
This was done in anticipation of the possibility of **importing** rooms directly from the interface.

Before creating a room, it is checked that the chosen URL is **not malformed** and if this check fails, the room is not created.
Finally, the additional privileges for streaming, annotations, and 3D models will be by default the same as the chosen access level, in order to simplify the process of adding a room.

3.6.5 Room deletion



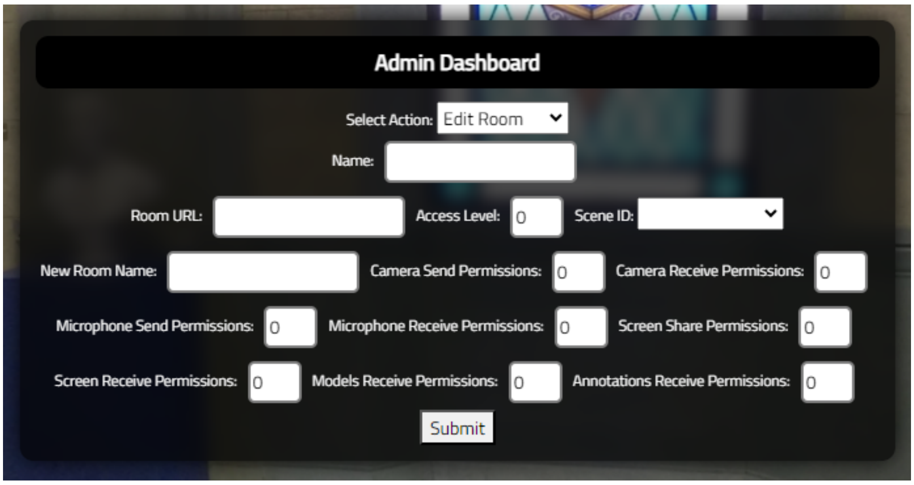
Select Action: Delete Room

Figure 3.9: Administrator dashboard for room deletion

By choosing "**Delete Room**" and entering the name of a room, it will be deleted from the database.

In case the deleted room was attended by users at the time of deletion, they will be **redirected** to the landing page and informed of what happened by an alert.

3.6.6 Room Edit



Select Action: Edit Room

Access Level: 0

Camera Send Permissions: 0

Camera Receive Permissions: 0

Microphone Send Permissions: 0

Microphone Receive Permissions: 0

Screen Share Permissions: 0

Screen Receive Permissions: 0

Models Receive Permissions: 0

Annotations Receive Permissions: 0

Figure 3.10: Administrator dashboard for room editing

The last option (“**Edit Room**”) allows an administrator to edit a room.
By entering the name of a room, it will be possible to change all its fields, including streaming permissions, annotations, and 3D models.

Again, the server will edit only the fields that have been filled in the form.

Starting with the room name and URL, if they are changed they should remain unique so that rooms can be identified. Malformed URLs are still not allowed.

If a name or URL is changed to a room while it is frequented by users a **reload** of the page or a **redirect** to the new room URL will occur, respectively.

If the scene ID of a room is changed while it contains users, the page will be **reloaded** to reflect the change.

If a change in the access level of a room with users in it occurs, if the new required level has become greater than that of one of the users, that user will be **redirected** to the landing page.

Finally, regarding media stream permissions, they will update and streams will be banned or allowed **dynamically**, both for sending and receiving.
3D models and annotations will require a reload of the page as explained in 3.6.3.

3.7 Logs

All operations indicated in section 3.6, along with login, logout and JWT token refreshes are recorded in a dedicated `security_log.csv` **log file**.
The format chosen for the logs is `.csv` given its versatility and ease of consultation.

The logs are recorded with their timestamps, both in UNIX and human readable format, in the form USERNAME - ACTION - OBJECT.
Login, logout and tokens refresh do not have OBJECT, in which case the field remains blank.

Actual examples of logs are the following:

```
1718479423787,2024-06-15T19:23:43.787Z,admin,LOGGED IN,
1718479821868,2024-06-15T19:30:21.868Z,admin,EDITED USER,testUser
```

3.8 Beyond virtual tour

The latest implementation did not strictly address the area of security and privacy but leveraged previously defined access permissions to try to provide a user experience beyond the virtual tour interface.

Limited to room “room2,” which is where the 3D model of the St. Cristina complex is rendered, users are shown a **minimap** in the upper right corner of the screen.

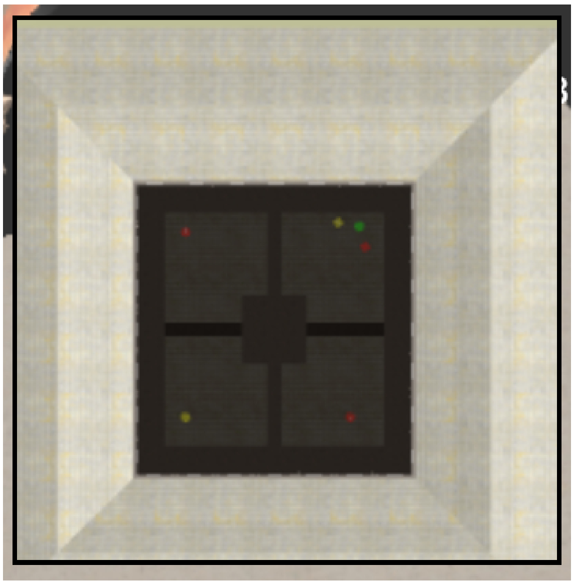


Figure 3.11: Minimap

Administrators, by clicking on a point on the minimap, will be able to select any of the existing rooms and place a **beacon** on that point that if passed through will teleport to the chosen room.

Clearly, beacons are designed to respect access rights to rooms and will appear **colored** red if the user cannot access the room. In this case user will not be teleported to the room.
On the other hand, yellow will be shown in case access is allowed but at least one of the eight additional permissions is insufficient.
Otherwise a green beacon will be shown.

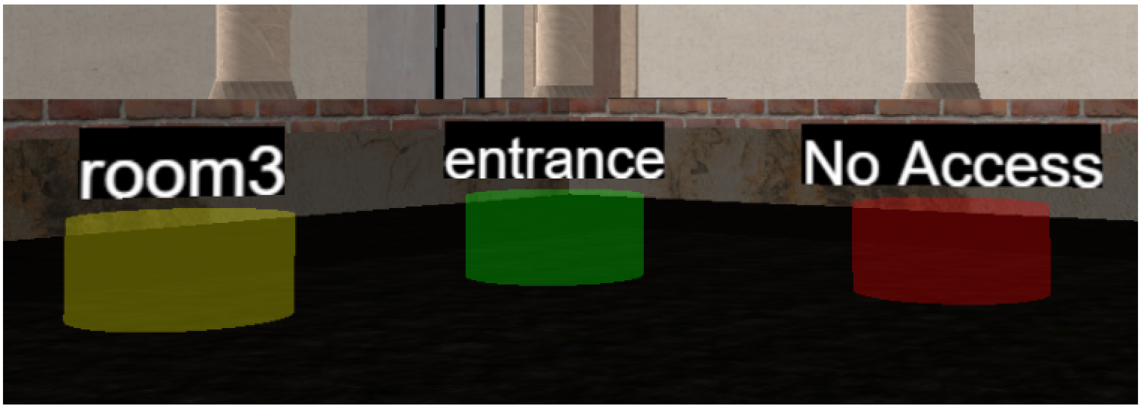


Figure 3.12: Colored beacons

3.9 Locations of the changes

This section will summarize where the previously mentioned changes can be found within the code.
For a better understanding it is possible to refer to Figure 2.14.

Regarding the new APIs, both for authentication (§3.1) and for managing rooms and users (§3.2 - §3.3) and their logs (§3.7), the `newAPI.js` file was created and `Core.js` was integrated with the necessary functions.

Dynamic routes (§3.2) have been added to `ATON.service.main.js`.

The virtual tour interface (§3.4) was made dynamic by modifying the `SUI.panoSelectorToolbarSetup` function in `ATON.sui.js`.

Permissions for streaming, 3D models and annotations (§3.5) were implemented by modifying `ATON.mediaflow.js`, event handlers in `ATON.photon.js` and `ATON.scenehub.js`.

The user interface of the admin dashboard (§3.6) was implemented by creating the `const formTemplate` and the associated logic in `hathor.ui.js`.
Event handlers related to user and room modifications have been implemented in `editEvents.js`.

Finally, everything related to minimap and beacons (§3.8) can be found in `ATON.miniMap.js`.

The code will be available once the platform and the repository will be public at the following link: **https://github.com/Sc1anso/AugmentedAlma**.

Chapter 4

Discussion and future work

This thesis work focused on **improving security and privacy** of the Augmented Alma platform.

Specifically, a system was implemented to include **access rights** to the different rooms that can be entered by users and, within these, a system that further differentiates users based on the possibility of creating or receiving audio or video streams and seeing spatial annotations or 3D models that are present in the virtual environment.

This differentiation of users is also **new** for collaborative extended reality applications and allows administrators, who in a wider context might be seen as the organizers or co-organizers of meetings, to have as much control as possible over what individual users in a room can do or see.

In an educational context or similarly during meetings attended by many guests this firstly ensures **order** by preventing everyone from initiating audio-video streams at the same time and secondly ensures that those who are not authorized either to access the room in which the meeting is being held or to view others' streams or 3D models and spatial annotations **cannot receive** any kind of sensitive information from them.

Even in a context in which the platform would be used for meetings among few people these features would still remain useful in case some information needs to be secreted only for a limited period of time or from a certain moment in the meeting onwards.

This differentiation of users could also be useful in enforcing **privacy policies**, as suggested by Marques et al. (2023), since for users who do not wish to share their data, such as voice or camera, with third parties or with service provider platforms

it could simply be inhibited the access to these streams so that they cannot even be initiated.
This would be another method besides the standard permissions that the operative systems of the devices often present to allow users to freely choose how to protect their privacy.

To conclude, an **assessment** of the impact this work had on security and privacy will be carried out from what is shown in Tables 1.1 and 1.2.

- Regarding **integrity**, no special measures were taken to prevent users from interfering with the virtual environment, however, thanks to access permissions a system could be implemented that prevents untrusted users from interacting with the environment so it is possible to say that the **groundwork** was set to strengthen this property.
- For **non-repudiation**, the logs system explained in Section 3.7 was implemented.
- **Availability** depends on the server and not specifically on this work, however all the implemented code contains error handling routines to try to minimize fatal errors.
- For **authorization** and **access control**, the new authentication system explained in Section 3.1 and the differentiation system that controls which users can perform which actions have been implemented.
  As mentioned earlier, the system can be **expanded** to include any action.
- On **identification**, after authentication, JWT tokens, which are unique to each user, are used.
- **Confidentiality** is guaranteed by the user differentiation system: only authorized users can access the information in the rooms.
- **Anonymity** or **pseudonymity** is currently guaranteed through the use of usernames but in other contexts, such as teaching or business meetings, this may not be a desired property.
- **Unlinkability** and **plausible deniability** for this platform are undesirable since priority has been given to the security property of non-repudiation and these properties represent the **opposite**.
- As for **undetectability** and **unobservability**, for unauthorized users, this is limited to streaming, 3D models, and annotations.
- **Content awareness** and **consent compliance** will definitely be implemented once the platform is complete.

These will certainly not be the final implementations of all the properties, but certainly all the necessary ones have been **strengthened** by this work.

4.1 Further improvements

Limited to this thesis work, some improvements remain possible.

First, regarding events related to room and user changes, it would be possible to move some of the logic delegated to clients **to the server** so as to further relieve user devices.
This is not particularly resource-intensive code however given the goal of running on as many devices as possible, even non-top-of-the-line ones, lightening the work of the clients as much as possible would be beneficial.

Another possible improvement would be to implement a system to **upload custom** 360-degree images or models when creating a room, as anticipated in Section 3.6.4 or otherwise allow each organization to have its own custom models.

A third improvement, especially with regard to the VR interface, would be to give administrators the ability to act on users by **clicking** on them and having an interface to change their parameters so that they do not necessarily have to use the dashboard.

Sticking with the user interface, creating a **registration** interface, as introduced in Section 3.6.1, could also be useful, although it is not the only existing registration method and therefore could be replaced or integrated with others.

A last possible implementation could be the inclusion of a user to act as a “`root`” i.e., the super-user who can assign and remove the administrator role and access rights without having them taken away.

4.2 Testing

All new features were tested locally on Windows 10 and 11 systems on Chrome, Edge, and Firefox browsers with up to six users per collaborative session.

However, **subsequent** tests are planned after deployment on server.

4.3 Use of LLMs

During code writing, large language models were used as an aid to code development, specifically **GPT 4o** and **GPT 3.5** for textual answers and **DALL-E 3** for images.
The following are the code points and prompts used:

4.3.1 Admin dashboard form

To create the HTML template for the form that realizes the administrator dashboard, this form is in the `const formTemplate` in `/public/hathor/hathor.ui.js`.
The following prompt was employed:

> Create an HTML form template that allows users to add, edit, and delete both users and rooms.
> The form should have a drop-down menu to select the action (add, edit, delete) for users and rooms.
>
> For user actions:
> - Add User: Requires fields for the user's name, password, and privilege level.
> - Edit User: Requires fields for the username, new username, password, privilege level, and Sketchfab token.
> - Delete User: Requires only the user's name.
> In this case privilege level is a number from 1 to 5.
>
> For room actions:
> - Add Room: Requires fields for the room name, URL, access level (0-5), and scene ID (a drop-down menu with 10 options).
> - Edit Room: Requires fields for the new room name, URL, access level, scene ID and permissions for camera send/receive, microphone send/receive, screen share/receive, models receive, and annotations receive (all with values from 0-5).
> -Delete Room: Requires only the room's name.
> Add a submit button that triggers a function to handle the form submission.

The template thus generated was **adapted** to the specific needs of the platform and was provided with the functions for submitting.

4.3.2 Minimap example

Before working on the minimap functionality shown in Section 3.8, the artificial intelligence was asked to generate a **standalone example** with the minimum functionalities needed.
The prompt used was the following:

> Create an example HTML file with related script "`minimap.js`" in which the file "`./file.glb`" containing a 3D model is rendered.
>
> Add a minimap in the upper right corner showing the model from above. When the minimap is clicked, convert the click coordinates to world coordinates and at the corresponding position place the center of a colored circle with radius 5 and render it.

From this minimal working example, the code was **rewritten** to integrate existing elements, such as the ATON camera, to turn circles into red, yellow, or green beacons depending on access permissions (see Figure 3.12) and make requests to the server to add or delete circles.
It is also checked whether the user is in a beacon to teleport them in case of sufficient permissions.

This code can be found in the file "`ATON.miniMap.js`.

4.3.3 JSDoc documentation

For some particularly long functions or those involving many cases, LLM was used to generate **JSDoc-style documentation**[1] with the prompt:

> Generate JSDoc documentation for the following function:

and the function below.

The generated documentation was then checked and corrected if necessary.

[1] JSDoc website: https://jsdoc.app/

4.3.4 Dashboard icon

The administration dashboard icon was generated with the following prompt:

> Create using only white on a black background an icon for an administration dashboard from which the possibility of changing various parameters is understandable.

The result was the following:

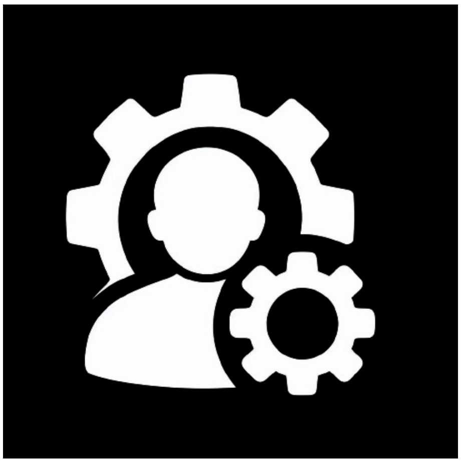


Figure 4.1: Admin dashboard icon

This image had its background removed and has been included in the Augmented Alma icons set.

Bibliography

Al Hafidz, I. A., Sukaridhoto, S., Al Rasyid, M. U. H., Budiarti, R. P. N., Mardhotillah, R. R., Amalia, R., … Satrio, N. A. (2021). Design of collaborative webxr for medical learning platform. In *2021 international electronics symposium (ies)* (pp. 499–504).

Butz, A., Beshers, C., & Feiner, S. (1998). Of vampire mirrors and privacy lamps: Privacy management in multi-user augmented environments. In *Proceedings of the 11th annual acm symposium on user interface software and technology* (pp. 171–172).

De Guzman, J. A., Thilakarathna, K., & Seneviratne, A. (2019). Security and privacy approaches in mixed reality: A literature survey. *ACM Computing Surveys (CSUR)*, *52*(6), 1–37.

Delgado, D. A., & Ruiz, J. (2024). Evaluation of shared-gaze visualizations for virtual assembly tasks. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 765–766).

Fanini, B., Ferdani, D., Demetrescu, E., Berto, S., & d'Annibale, E. (2021). Aton: An open-source framework for creating immersive, collaborative and liquid web-apps for cultural heritage. *Applied Sciences*, *11*(22), 11062.

Fiore, M., Gattullo, M., Mongiello, M., & Uva, A. (2024). Merging blockchain and augmented reality for an immersive traceability platform. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 933–934).

George, A., & Routray, A. (2016). A score level fusion method for eye movement biometrics. *Pattern Recognition Letters*, *82*, 207–215.

Giaretta, A. (2022). Security and privacy in virtual reality–a literature survey. *arXiv preprint arXiv:2205.00208*.

Giunchi, D., Bovo, R., Numan, N., & Steed, A. (2024). Streamspace: A framework for window streaming in collaborative mixed reality environments. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 859–860).

Guo, Q., Fu, J., Lu, Y., & Gan, D. (2024). Diffusion attack: Leveraging stable diffusion for naturalistic image attacking. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 975–976).

Happa, J., Glencross, M., & Steed, A. (2019). Cyber security threats and challenges in collaborative mixed-reality. *Frontiers in ICT*, *6*, 5.

Jones, M., Bradley, J., & Sakimura, N. (2015). *Rfc 7519: Json web token (jwt).* RFC Editor.

Kanaoka, A., & Isohara, T. (2024). Enhancing smishing detection in ar environments: Cross-device solutions for seamless reality. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 565–572).

Kasprowski, P., & Ober, J. (2004). Eye movements in biometrics. In *International workshop on biometric authentication* (pp. 248–258).

Korečko, Š., Hudák, M., Sobota, B., Sivỳ, M., Pleva, M., & Steingartner, W. (2021). Experimental performance evaluation of enhanced user interaction components for web-based collaborative extended reality. *Applied Sciences*, *11*(9), 3811.

Kürtünlüoğlu, P., Akdik, B., & Karaarslan, E. (2022). Security of virtual reality authentication methods in metaverse: An overview. *arXiv preprint arXiv:2209.06447*.

Li, J., Arora, S. S., Fawaz, K., Kim, Y., Liu, C., Meiser, S., . . . Wagner, K. (2024). Exploring the interplay between interaction experience and security perception of payment authentication in virtual reality. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 1043–1044).

Lohr, D., & Komogortsev, O. V. (2022). Eye know you too: Toward viable end-to-end eye movement biometrics for user authentication. *IEEE Transactions on Information Forensics and Security*, *17*, 3151–3164.

Lv, Z., Chen, D., Lou, R., & Song, H. (2020). Industrial security solution for virtual reality. *IEEE Internet of Things Journal*, *8*(8), 6273–6281.

Marques, B., Silva, S., & Santos, B. S. (2023). Where do we stand on ethics, privacy, and security for scenarios of remote collaboration supported by extended reality? In *2023 ieee international symposium on mixed and augmented reality adjunct (ismar-adjunct)* (pp. 355–359).

Martí-Testón, A., Muñoz, A., Gracia, L., & Solanes, J. E. (2023). Using webxr metaverse platforms to create touristic services and cultural promotion. *Applied Sciences*, *13*(14), 8544.

Mathis, F., Fawaz, H. I., & Khamis, M. (2020). Knowledge-driven biometric authentication in virtual reality. In *Extended abstracts of the 2020 chi conference on human factors in computing systems* (pp. 1–10).

Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, *77*(12), 1321–1329.

Nair, V., Guo, W., O'Brien, J. F., Rosenberg, L., & Song, D. (2023). Deep motion masking for secure, usable, and scalable real-time anonymization of virtual reality motion data. *arXiv preprint arXiv:2311.05090*.

Nair, V., Rack, C., Guo, W., Wang, R., Li, S., Huang, B., … others (2024). Inferring private personal attributes of virtual reality users from ecologically valid head and hand motion data. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 477–484).

Pearson, J., Robinson, S., Jones, M., Joshi, A., Ahire, S., Sahoo, D., & Subramanian, S. (2017). Chameleon devices: investigating more secure and discreet mobile interactions via active camouflaging. In *Proceedings of the 2017 chi conference on human factors in computing systems* (pp. 5184–5196).

Pereira, V., Matos, T., Rodrigues, R., Nóbrega, R., & Jacob, J. (2019). Extended reality framework for remote collaborative interactions in virtual environments. In *2019 international conference on graphics and interaction (icgi)* (pp. 17–24).

Ramirez, G. N., Spivack, J., & David-John, B. (2024). Deceptive patterns and perceptual risks in an eye-tracked virtual reality. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp.

341–344).

Rupp, D., GrieBer, P., Bonsch, A., & Kuhlen, T. W. (2024). Authentication in immersive virtual environments through gesture-based interaction with a virtual agent. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 54–60).

Slocum, C., Zhang, Y., Abu-Ghazaleh, N., & Chen, J. (2023). Going through the motions:{AR/VR} keylogging from user head motions. In *32nd usenix security symposium (usenix security 23)* (pp. 159–174).

Slocum, C., Zhang, Y., Shayegani, E., Zaree, P., Abu-Ghazaleh, N., & Chen, J. (2023). That doesn't go there: Attacks on shared state in multi-user augmented reality applications. *arXiv preprint arXiv:2308.09146*.

Stacchio, L., Vallasciani, G., Augello, G., Carrador, S., Cascarano, P., & Marfia, G. (2024). Wixard: Towards a holistic distributed platform for multi-party and cross-reality webxr experiences. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 264–272).

Sun, R., Wang, H., Xue, M., & Chen, H.-T. (2024). Ppvr: A privacy-preserving approach for user behaviors in vr. In *2024 ieee conference on virtual reality and 3d user interfaces abstracts and workshops (vrw)* (pp. 1055–1056).

Wilson, E., Ibragimov, A., Proulx, M. J., Tetali, S. D., Butler, K., & Jain, E. (2024). Privacy-preserving gaze data streaming in immersive interactive virtual reality: Robustness and user experience. *arXiv preprint arXiv:2402.07687*.

Zhang, Y., Slocum, C., Chen, J., & Abu-Ghazaleh, N. (2023). It's all in your head (set): Side-channel attacks on ar/vr systems. In *Usenix security.*

Acknowledgements

I would like to dedicate this last part to acknowledgements.

First, I sincerely thank my supervisor, Prof. Donatiello and my co-supervisor, Prof. Marfia for following and advising me during the writing of this thesis.
A special thanks to Giacomo Vallasciani for his support and all the explanations and advice he gave me.

I also jointly thank the University of Bologna and Bielefeld University and their Professors for giving me the life-changing opportunity to carry out the double degree program.
My gratitude goes in particular to Prof. Martini, Federico Pennino and Atahualpa for the support and the helpful advice they gave me while I was in Bielefeld.

Thanks to all my friends, both the new ones I met on Erasmus and the ones I have known for a longer time.

Finally, my deepest gratitude goes to my extended family, to whom this thesis is dedicated for the great support and love they have given me over the years.