D5.1

SUN Platform Architecture

Project Name:	Social and hUman ceNtered XR
Project Acronym:	SUN
Project No.:	101092612
Call:	HORIZON-CL4-2022-HUMAN-01
Topic:	HORIZON-CL4-2022-HUMAN-01-14
Type of Action:	HORIZON-RIA
Service:	CNECT/G/02
Start of Project:	1 December 2022
Duration:	36 months
Project Website:	www.sun-xr-project.eu

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101092612

www.sun-xr-project.eu

Deliverable Information

Deliverable title	Sun Platform Architecture
Deliverable number	D5.1
Deliverable version	V1.0
Previous version(s)	V0.5
Contractual delivery date	31 May 2024
Actual delivery date	28 May 2024
Deliverable type	Report
Dissemination level	Public
Work Package	WP5: Integrated XR Platform
Task(s)	T5.1
Partner responsible	ENG
Author(s)	Ferdinando Bosco, Giovanni Di Marco (ENG) Alexandru Stan (IN2) George Loukas (UoG) Panagiotis Kasnesis, Lazaros Toumanidis (TG) Ioannis Paraskevopoulos (IG) Vincent Mendez (EPFL) Leesa Joyce (HOLO) Claudio Vairo (CNR) Spyridon Symeonidis, Sotiris Diplaris, Vasileios-Rafail Xefteris, Ilias Poulios, Panagiotis Vrachnos, Georgios Loupas, Orestis Sarakatsanos (CERTH)
Editor	Ferdinando Bosco (ENG)
Reviewers	Claudio Gennaro (CNR), Katerina Mania (TUC)
EC Project Officer	Adelina Cornelia Dinu

Total number of pages	114
Abstract	This document reports the SUN Architecture design and technical specifications for implementing the SUN XR Platform.
Keywords	XR Platform, XR Reference Architecture, Technical Specifications

Disclaimer. *Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.*

Revision History

VERSION NO.	DATE	AUTHOR/REVIEWER	MODIFICATION
0.1	05/03/2024	Ferdinando Bosco (ENG)	Table of Content
0.2	19/04/2024	Ferdinando Bosco, Giovanni Di Marco (ENG) Alexandru Stan (IN2) George Loukas (UoG) Panagiotis Kasnesis, Lazaros Toumanidis (TG) Ioannis Paraskevopoulos (IG) Vincent Mendez (EPFL) Leesa Joyce (HOLO)	Section 2,3 and 4 concluded. Section 5 and 6 First draft
0.3	30/04/2024	Ferdinando Bosco (ENG) Panagiotis Kasnesis (TG) Claudio Vairo (CNR)	Components list on Section 4 updated. Section 5 and 6 concluded.
0.4	03/05/2024	Ferdinando Bosco (ENG)	Draft ready for internal review
0.5	27/05/2024	Ferdinando Bosco (ENG)	Updated version based on reviewer comments
1.0	28/05/2024	Giuseppe Amato (CNR)	Submitted version

Peer Reviewers

NAME	ORGANIZATION
Claudio Gennaro	CNR
Katerina Mania	TUC

Table of Abbreviations and Acronyms

ABBREVIATION	DESCRIPTION
API	Application Programming Interface
AR	Augmented Reality
EMG	Electromyography
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMD	Head Mounted Display
Industry 5.0	An evolution of the Industry 4.0 concept (i.e. intelligent networking of machines and processes for industry with the help of ICT) where even more advanced technologies will be integrated into industrial processes for further optimization and efficiency gains
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
MR	Mixed Reality
OV	Open Vocabulary
P2P	Peer-to-Peer
REST	Representation State Transfer
RPC	Remote Procedure Call
s-EMG	Surface electromyography
SDK	Software Development Kit
TCP	Transmission Control Protocol
TRL	Technology Readiness Level
UDP	User Datagram Protocol
UI	User Interface
VR	Virtual Reality

WP	Work Package
XR	Extended Reality

Table of Contents

List of Figures

List of Tables

1 Executive Summary

The design phase of the SUN Project mainly focused on the definition of pilots, scenarios, and user requirements. Starting from them, it was fundamental to define the overall system architecture and technical specifications, to prepare the implementation and integration phase.

The methodology adopted and described in this document, allowed us to design the SUN Reference Architecture as a high-level architecture able to provide guidance for technical developments, incorporating the vision of the solution, as well as the requirements and the technical specifications. It can be intended as the shared baseline for implementing all the possible systems that will be part of the SUN XR Platform.

The iterative and collaborative approach allowed us to consider the feedback and considerations of both technical and user partners, in order to design a system that can satisfy both the technical and user expectations to be validated in the SUN XR project.

The result described here consists of the SUN Reference Architecture, as well as the list of components to be implemented and integrated within the different releases of the SUN Platform, with related interfaces and interactions, mapped in different pilot and use cases.

2 Introduction

2.1 Scope

The main objective of the WP5 is to design and implement the overall SUN Platform. The platform aims to create a modular and configurable system that enables extended reality (XR) features, blending the physical world and the virtual one and offering those innovative processes to the users.

As a preliminary activity for implementing the SUN Platform, it was necessary to define the Reference Architecture and develop the technical specifications.

2.2 Task T5.1

T5.1 main goal is to define the high-level system architecture and technical specification of the SUN XR Platform.

Starting from the analysis of WP2 about user requirements and scenarios, together with the components' specification elicited in WP3 and WP4, T5.1 provided as outcome the SUN Reference Architecture and several technical specifications needed for implementing the integrated SUN XR Platform, able to be deployed and validated within all the SUN Pilots scenarios.

2.3 Outline of the deliverable

Section 3 of the document analyses the XR Platforms state of the art, including the most relevant initiatives and architectures, and putting them on the SUN context.

Section 4 described the methodology adopted for the design of the SUN Reference Architecture as well as for the definition of Technical Specifications for SUN XR Platform implementation.

Section 5 describes the SUN Reference Architecture itself and how it is mapped and adopted within the different SUN pilot scenarios, highlighting the SUN components utilisation and interactions.

Section 6 reports the identified technical specifications in terms of communication, integration, deployment, and security aspects.

Finally, Section 7 concludes this document.

3 XR Platform Architectures – State of Art

3.1 Introduction

Over the past decade, extended reality (XR) technologies, including Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR), have evolved from niche specialist tools to pivotal industry mainstays. They now play integral roles in sectors such as car manufacturing, healthcare education, architectural design, and pedagogy, among others. The momentum now leans towards interactive, multi-participant XR environments. Through local and cloud-based networks, the software can now sync data across various XR devices. Both tech giants and emerging social platforms are championing this evolution as the "Metaverse", positing it as the subsequent monumental phase in XR. This shift aims to amplify interpersonal engagements and enhance cooperative experiences in diverse XR settings.

3.2 Main initiatives and relevant architectures

In this chapter, we will present the architectures of XR platforms implemented in various use cases as documented in the literature.

3.2.1 AROS XR platform

From a study of occupational accidents, occupational safety experts have identified inadequate training and repetitive action as the main causes contributing to the accidents of workers using modern electromechanical machines and systems. AROS is based on augmented reality (AR) technologies and looks to help in reducing error rates. A system implemented on mobile devices allows you to project augmented reality instructions directly to the workplace. Through this AR system, workers are guided step by step in the execution of operational and safety procedures (Tatić & Tešić, 2017).

The architecture, presented in Fig. 1 is composed of two main components: the database and the application. The application is installed on a mobile device that acts as an interface for both system input and output. All necessary data is stored in a database that the application accesses as needed. When the system boots, the access subsystem is activated on the mobile device interface. The worker ID is used to query the Activity List table in the database, which assigns specific tasks to the user. Once a task is selected, the system retrieves the first set of instructions from the Instruction List, along with all the necessary parameters for tracking, and activates the augmented reality module. The system is, therefore, able to recognize a marker positioned in the industrial area. This marker activates the visualization of the operating and job management instructions in a virtual format while tracking the marker position. After tracking, a check is required to confirm the completion of the instructions. A positive confirmation activates the data storage of the completed task in the List of Finished Tasks (Tatić & Tešić, 2017).

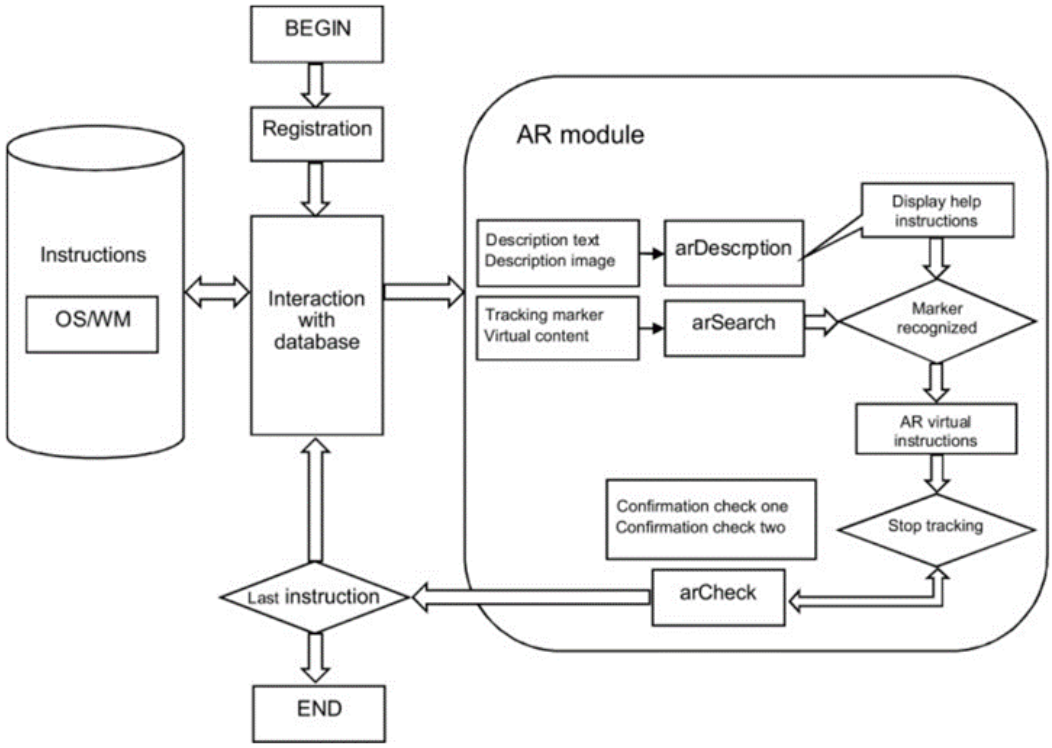

Figure 1: AR module. Tatić & Tešić (2017)

The database, hosted on a server, includes the Task List, the Instructions List, and the completed Tasks List. To assign an activity, the unique ID of a worker is read, which not only assigns the activity but also determines the volume of information displayed based on the worker's professional skills, as deduced from the codes attached to the instruction sets. After a task finishes, the completed task details are recorded in the Completed Tasks List. The Augmented Reality (AR) system described is implemented on portable devices, functioning independently as a mobile occupational safety system. This AR system, called AROC, is developed using the software Unity 3D and the Vuforia SDK and has been implemented and tested on the Android software platform (Tatić & Tešić, 2017).

3.2.2 Cross-platform XR structure for collaborative use

To effectively assess the capabilities of multi-user and cross-platform XR experiences, as noted by Tümler et al. (2022), the systems used must allow multiple users to interact simultaneously within the same virtual environment using various XR devices in real-time. Performance capabilities vary across platforms: PCs generally offer the most rendering and processing power. Although it is practical to run the same virtual scene on different platforms up to a certain level of data complexity, the specific devices used for this search include the Google Pixel 3 for smartphone-based interactions, a PC-VR setup with the configuration recommended by AltspaceVR, and the Microsoft HoloLens 2 as a wearable device on the head, reflecting the current industry standard. The Ubi-

Interact framework, although designed for distributed and responsive applications, is currently considered too complex for our needs(Tümler et al.,2022).

The table in Fig. 2 shows the compatibility of devices with the platforms.

Platform	Supported devices
OpenXR (Unity 2020.3.8+)	Microsoft HoloLens 2 Windows Mixed Reality headsets
Windows Mixed Reality	Microsoft HoloLens Microsoft HoloLens 2 Windows Mixed Reality headsets
Oculus (Unity 2019.3 or newer)	Oculus Quest
OpenVR	Windows Mixed Reality headsets HTC Vive Oculus Rift
Ultraleap Hand Tracking Mobile	Ultraleap Leap Motion controller iOS and Android

Figure 2: Block chart for user's xR experience. Tümler et al. (2022)

From the point of view of an XR developer, each platform requires its specific software development kit (SDK). For the PC-VR platform, developers can use VRTK among other tools, while the Android AR platform employs ARcore. Apple's iOS devices use Arkit, and Microsoft HoloLens 2 uses MRTK. A unique feature of MRTK is its versatility; it not only supports HoloLens 2 but also enables deployment on PC-VR headsets, Android smartphones, iOS smartphones, and other platforms. MRTK can adapt an XR application to meet platform-specific requirements. However, it does not facilitate the simultaneous collaboration of multiple users across different XR platforms. Unfortunately, MRTK does not currently support the Google Cardboard VR platform. As a result, for this research, development for Google Cardboard VR has been ruled out, despite its popularity among students because of its convenience. (Tümler et al.,2022)

Data type	Data source	Update requirements
Screen/head position	Coming from all devices	Real-time
Gaze direction	Coming from all devices	Real-time
Position and pose of controllers	Coming from VR	Real-time
Position and pose of hands and fingers	Coming from HoloLens 2	Real-time
Origin of world coordinate system	Registered between all devices	Once per start
Position and pose of spawned 3D objects	Coming from all devices	Real-time (interactive spawn object)/Once (static objects)
State of variables and function calls	Coming from all devices	Real-time
3D-Mesh of the "real world" around selected AR users	Coming from one selected AR user	Not necessary in realtime

Figure 3: Data that must be shared between xR devices. Tümler et al. (2022)

Data transmission between users is essential to enable real-time interaction and collaboration within the same virtual scene. The table in Fig. 3 provides examples of shared and synchronized data, sources, and upgrade requirements, regardless of the platform used. To address the challenges of data transmission, the market offers SDKs such as Photon and Mirror, which have robust features and support various data transfer methods. However, these SDKs are either closed-source or complex to develop. Moreover, none of these SDKs support cross-platform XR collaboration by default. (Tümler et al.,2022)

Finally, in addition to the calculation performed directly on the device, you can take advantage of moving the rendering and the elaborations outside using a cloud-based architecture for remote rendering. This option represents a predictable and advantageous possibility to allow high-quality real-time rendering on low-performance remote devices. In such scenarios, it is also important to know and implement device-specific interaction metaphors and data-sharing methods. These personalized approaches enhance the user experience and ensure efficient communication and functionality across different device types (Tümler et al.,2022).

User input data varies between devices. For example, VR headsets can capture tracking data from controllers, while smartphones generally can't. In a multi-user XR environment, when 3D objects are manipulated within the scene, it is not enough to update only the new position and rotation for the active user. Sending this information in real-time is a constraint for maintaining a good user experience and as much as possible natural interactions. Next to that when a user activates functions this must be visible and active also to other users, this process is called remote procedure call (RPC). Ideally, this synchronization takes place with almost zero latency. Therefore, in a multi-user scenario, two main types of data must be synchronized: first, the object laying data and, second, the RPC data. The schema in Fig. 4 represents the interactions between the players: objects, avatars, and users (Tümler et al.,2022).

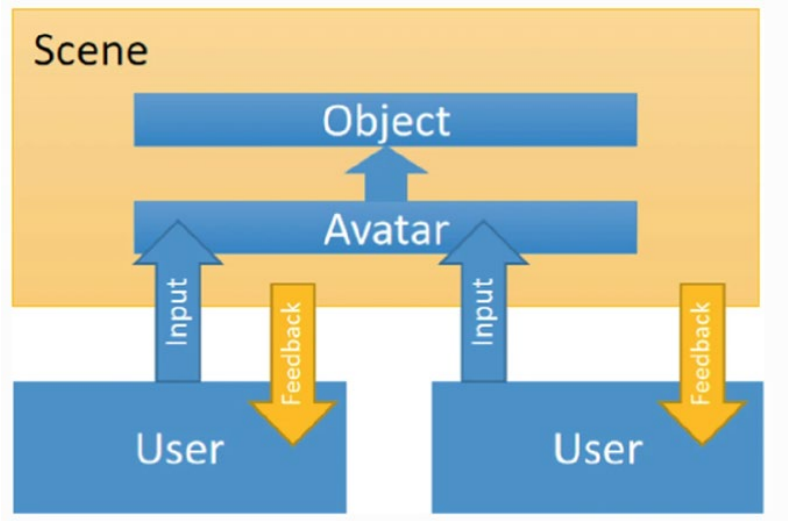

Figure 4: Block chart for user's xR experience. Tümler et al. (2022)

There are several basic network architectures, also schematized in Fig. 5, to exchange data between devices:

- Client-server model: the client(s) send(s) requests to the server that implements a functionality or a service and responds to them.
- N-tier model comes from the first model: "They divide different functions into different layers, and each layer performs its duties to complete the user's

request. This architecture is mostly used in complex systems". (Tümler et al., 2022)

- Peer-to-peer model (P2P): this architecture is different from the previous two, moreover it emphasizes that no device provides services, but all the devices work with each other as a peer. Peers can assume both roles as clients and as servers.

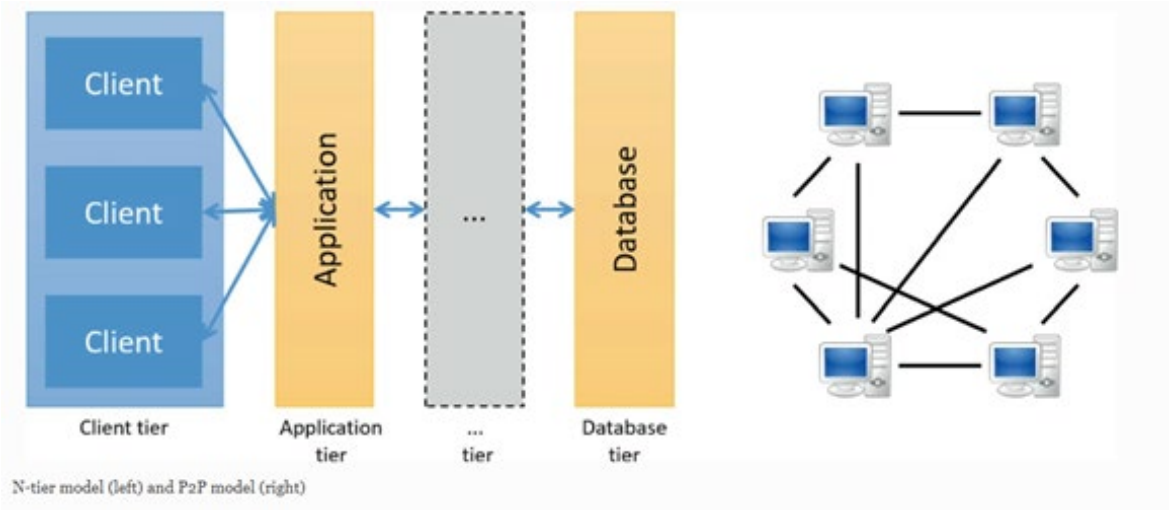

Figure 5: Block chart for user's xR experience. Tümler et al. (2022)

3.2.3 COVE-VR platform architecture

This research explores how Virtual Reality (VR) can be used to support asynchronous collaboration between geographically distributed departments engaged in developing maintenance methods and creating documentation (Burova et al. ,2022). The platform design aims to support asynchronous collaboration between two global teams located in different time zones, like cloud-based teamwork. Team members can leave comments and notes to others on their own or save and resume work later. The platform also allows team members to create digital content (such as photos, videos, and texts). The COVE-VR platform (shown in Fig. 6) includes two virtual environments (VE) and eight virtual tools to assist in identified industrial scenarios. All content generated in VR is saved to the hard disk folder and can be accessed later, easily integrating with standard office tools and other applications. COVE-VR implements a client-server model (discussed before) in which either synchronous or asynchronous collaboration is managed in separate servers. It includes a VR Client, a restful web service, a model converter developed internally, and a commercial component, the PUN server (Photon Unity Networking).

The platform has been developed with the Unity game engine and the Virtual Reality Toolkit (VRTK) 3.3.0, the VR Client includes essential components for a VR application, such as:

- renderer
- physical engine
- scripting runtime
- visual editor
- input system
- 3D model importers
- cross-platform support and components for VR user interfaces.

To synchronize environmental activities between multiple users during a session they use PUN while restful web service bridges its gap retaining the status of the virtual environment between sessions, since PUN does not support long-term data persistence. The most important software architectural decisions concerned serialization, the process of converting objects and data structures for transmission over the network or saving to files. They chose Odin Serializer, which allows you to directly serialize most of Unity's standard objects. This reduced the computational effort needed to implement snapshot synchronization and saving, especially when the default serialization behavior was sufficient (Burova et al. ,2022).

After their analysis regarding bugs and possible issues, Burova et al. (2022) pointed out that "most common bugs discovered were related to desynchronization, serialization failure, collision physics, and the effects of unanticipated user input, especially when multiple users affect the world state". Automated unit tests would probably not have revealed this type of bug, except for failures in serialization. Finally, Burova et al. (2022) stated, "it is a matter for future work to explore how user input should be simulated for automated tests; they are not aware of an existing test framework that supports VR user input simulation in Unity at this time."

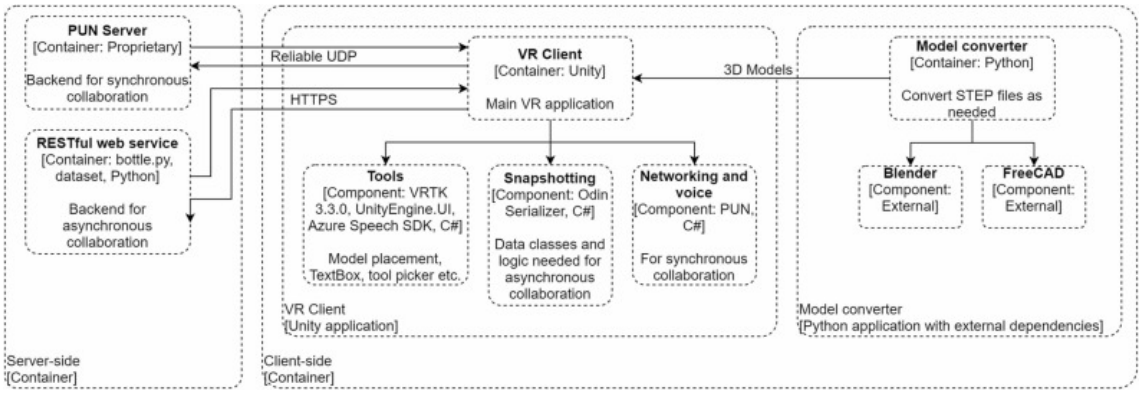

Figure 6: COVE-VR platform architecture. Burova et al. (2022)

3.2.4 Collaborative WebXR for medical training platform

This research, Hafidz et al. (2021), aims to support, by designing a collaborative WebXr platform, specialists in the medical field such as doctors and medical students. The goal is to collaborate (doctors, lecturers, and students) in the XR environment performing simulation in a way as much as possible near the real-time one.

Design

In this research infrastructure design (shown in Fig. 7), users can access WebXR via the Internet. The application of the high availability concept will affect the performance of the WebXR system. The load balancer in the infrastructure design serves as a vital component of the infrastructure used to increase the performance and reliability of WebXR by distributing workloads to other servers. The use of a load balancer and two web servers can minimize failure if the system takes a long time to access or does not connect due to many users trying to access the server simultaneously and cannot handle the traffic load. In this way, when one web server dies, the traffic will be diverted to another web server. The database server in this research infrastructure functions to store 3D assets, user credentials, the position of the last 3D object, and scenarios. Hafidz et al. (2021)

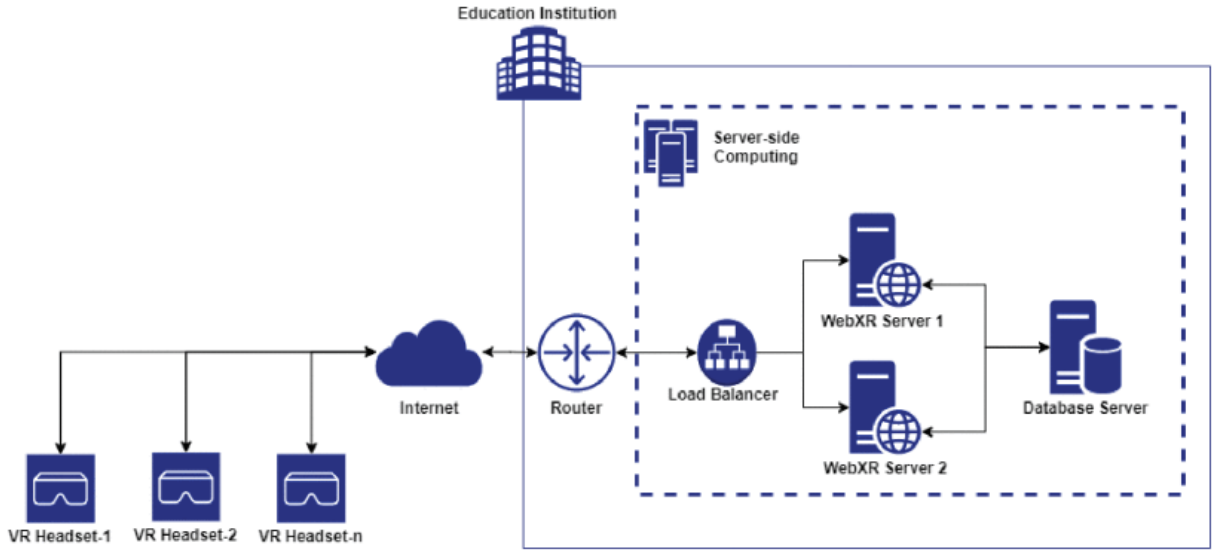

Figure 7: Infrastructre Architecture. Hafidz et al. (2021)

Based on the problems described previously, they designed a system based on a microservice approach. A server environment is divided into several separate services connected, as shown in Fig. 7. They use this approach because it has several advantages according to the system's characteristics in this research. These advantages include:

- Microservice applications are scalable, secure, and reliable. Each service can run simultaneously without disturbing other services.
- Independent process takes the advantage of modularity, especially in the maintenance process that gets easier. Since independent processes are less affected by cross interactions and cross dependencies.
- Can adapt to new technological developments.

They use two servers using containers to host medical learning platform applications. Containers are packages or applications that rely on virtual isolation to run applications that can run multiple operating system kernels simultaneously without the need for virtual machines (VMs). Containers provide overall flexibility compared to using physical servers and virtual machines. Containers can run directly on the Operating System (OS) without using a hypervisor. They use containers as opposed to VMs because the resource allocations for containers and VMs are different. Containers have resource allocations that can be shared directly by the Host Server itself. The container will take the resource allocation that is on the hardware according to what the container needs. They use the REST API and Publish-Subscribe Gateway for data connection between clients and XR scenarios that have been created on the medical learning platform. Hafidz et al. (2021).

They use a load balancer to distribute network traffic to two WebXR Servers. The load balancer ensures that one of the servers does not take on too many requests. The load balancer software they use is Nginx Load Balancer. A lot of engineers working in the field

have taken advantage of using Nginx as an HTTP Load Balancer (load sharer) to avoid webapp overload. When used as an HTTP Load Balancer, Nginx will manage the load based on the specified criteria. Not only HTTP, but Nginx can also do forwarding for TCP Load Balancer and UDP Load Balancer. They use the least connection method to manage the traffic load on the WebXR medical learning platform application. The least Connection Algorithm is a method that addresses the limitations of the round-robin algorithm in evaluating the load of each server. The Least Connection method maintains an even distribution of traffic across all available servers. If a server has a large connection load, the data requests will be distributed to the sparer servers. When a request occurs, Least Connection tries to distribute it to the server with the least number of connections. This is done to avoid overloading the server due to the large amount of traffic it receives. Hafidz et al. (2021)

Data Storage

All data related to scenarios and users will enter the database server. They use SQL database data storage for users and NoSQL database as data storage for XR scenarios. SQL and NoSQL have backward compatibility with the previously described system. This will make it easier for us to develop this medical learning platform. SQL is designed for relational database management systems, so it is suitable for data storage that stores user data. The main applications of SQL include writing integrated scripts, setting up and running query analysis, and adding, updating, and deleting rows and columns of data in a database. As for NoSQL, they can store large volumes of unstructured XR scenario data. The data storage architecture enhances the speed of read and write operations and supports horizontal scaling of servers. The infrastructure they design supports remote networks using the internet. So that users can access the content of the medical learning platform from anywhere and anytime, this fits perfectly with our goal of creating a flexible online learning alternative. Hafidz et al. (2021)

Multi-cross VR Platform

Based on the infrastructure described above, this work presents an initial stage development design for the WebXR platform, which will be integrated with VR devices, namely Oculus Quest 1, Oculus Quest 2, and Oculus Rift S. Development is carried out on the WebXR scene, which will display WebXR visualizations, Hafidz et al. (2021). In the WebXR scene, a WebGL engine functions to render a set of 3D objects on the web. The WebGL engine will develop several services, namely physics engine, mesh, camera view, lighting, GUI, and WebXR API. At this stage, the user will be able to interact directly with the virtual model freely and perform simulations according to the simulation scenario that has been built on the medical scenario stored in the database. Users can access this service through a Web Browser that supports WebXR technology on their respective devices. Then to support collaborative and multiuser features, a gateway server will be used, which can synchronize 3D in the virtual environment that has been created. Every transfer of 3d objects, made in real-time, is translated into publish and subscribe from

the gateway server to all the users. With the WebXR API feature on the WebGL engine, this service can be integrated with VR / MR devices, such as Oculus, Magic Leap, and others.

3.2.5 Bwell: an immersive platform for cognitive evaluation and rehabilitation

Immersive technologies such as virtual reality can enable clinical care that aligns significantly with cognitive deficits in the real world. Nevertheless, there are not so many available options. This is due to the challenges of developing in a sensitive field such as the medical one. Development is complicated by the need to integrate industry expert recommendations at all stages (Gagnon Shaigetz et al.,2021).

Platform key features

Key features of bWell include a multimodal (fully, partially, or not immersive) and multi-platform implementation (extended reality, mobile and PC), configurable exercises that combine standardized assessments with adaptive and playful variants for therapy, a user interface for therapists for task administration and dosing, and automatic recording of activity data (Gagnon Shaigetz et al.,2021). bWell was designed to serve as a widely applicable toolkit, which focuses on the general aspects of cognition commonly impacted in many disorders, rather than a single disorder or a specific cognitive domain. Neuropsychological assessments are currently changing, moving from traditional paradigms based on constructs and on pen and paper to tests representing everyday life, attracted using immersive technologies, also known as extended reality (XR). Virtual reality (VR) allows for almost complete sensory immersion with extensive design possibilities and rigorous experimental control, making it ideal for evaluating cognitive functioning in performing real-life simulated tasks. The ability of VR to provide and control stimuli while capturing high-fidelity responses during an exercise provides a controlled and repeatable tool that is not available in traditional test methods. Most VR platforms include a single exercise adapted to a specific disturbance, require manual reconfiguration of operation to support repeated measurements, and support limited and often specific hardware for user visualization and interaction.

Methods

To identify the key requirements for the platform a human-centred approach was followed ensuring that these 3 criteria are followed: desirable, viable, and feasible (see Fig. 8).

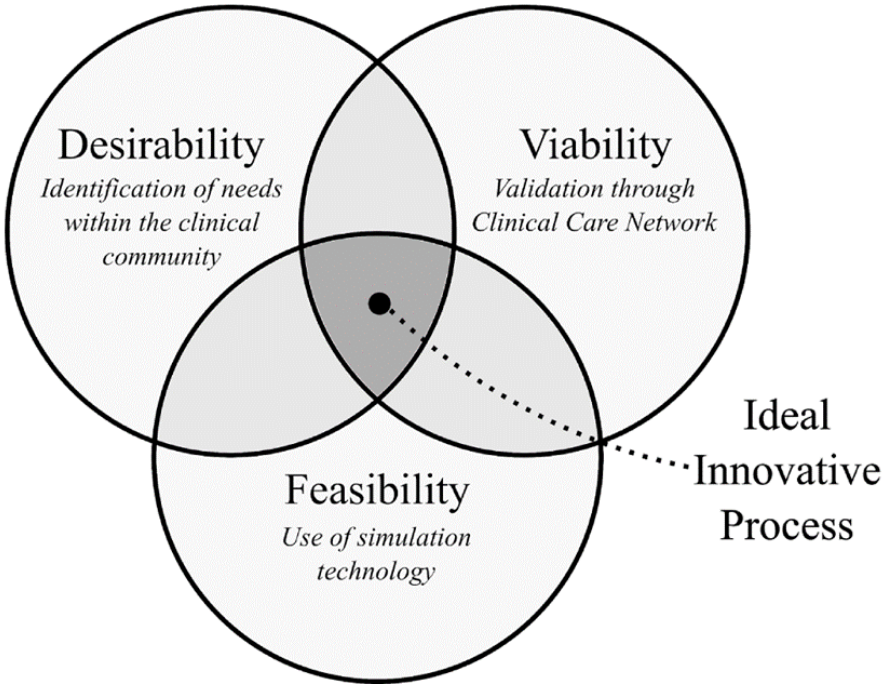

Figure 8: Venn diagram showing innovative solution sweet spot that lies at the intersection of desirability, viability, and feasibility. Gagnon Shaigetz et al. (2021)

The Level of Technological Readiness (TRL) framework has been used to integrate and structure the phases of clinical collaboration within the technological development phases according to maturity level. Considering the needs of cognitive care, technological possibilities, and innovation potential, four main requirements for the bWell platform have been identified:

- Support for multiple hardware platforms and different immersion modes.
- A suite of customizable tasks.
- A user interface for clinicians to control task parameters.
- A data logging mechanism.

Collaboration with clinical partners is a fundamental requirement. In the image in Fig. 9 there is a diagram that demonstrates the phases of clinical collaboration with the different TRL. (Gagnon Shaigetz et al.,2021)

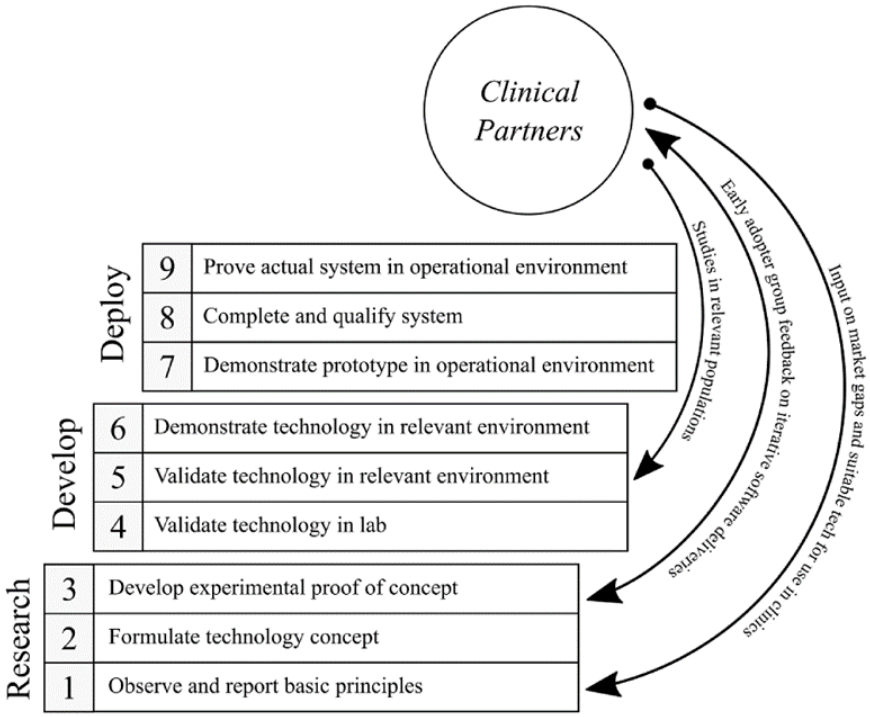

Figure 9: Schematic demonstrating the stages of clinical collaboration at the different TRLs. Gagnon Shaigetz et al. (2021)

Implementation

The implementation involved creating a platform supported by immersive technologies (Fig.10). This platform was developed by transforming the requirements collected during the active co-design process into hardware and software components, including content design.

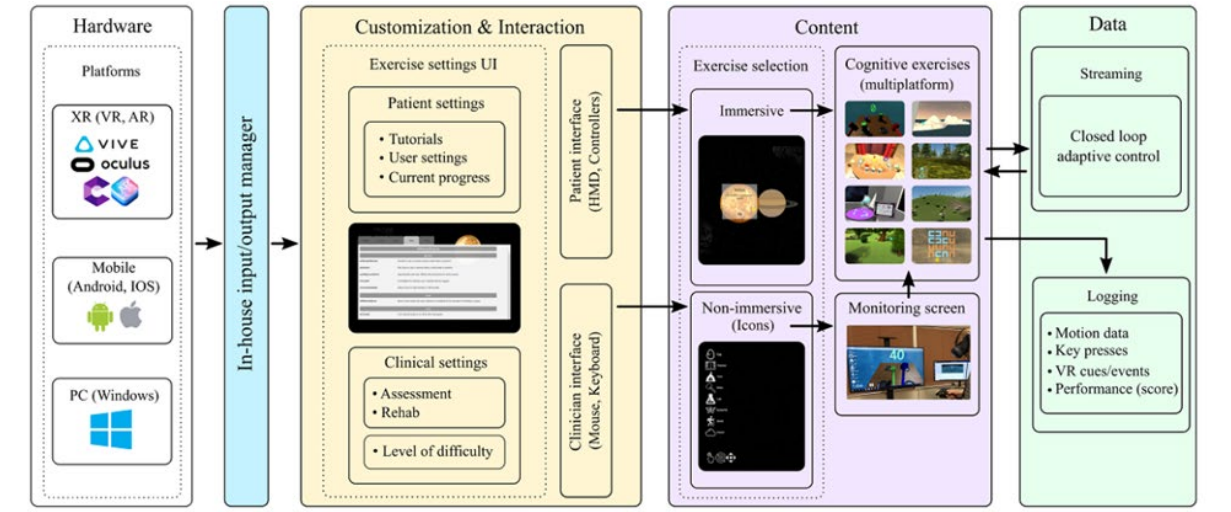

Figure 10: bWell architecture components. Gagnon Shaigetz et al. (2021)

Bwell was developed using the Unity 3D game engine and includes several components. The task development focused on a generic core that orchestrates the flow and interaction between various components (software and hardware). An internal input manager has been created to support different types of hardware like XR headsets. It acts as an abstract layer that assigns device inputs and outputs to specific software functions. Clinicians and patients interact with the platform through distinct interfaces to improve their interaction. In the beginning, clinicians access a non-invasive user interface to adjust test settings, customizing the surgery to the patient's needs. Patients, however, select exercises within a virtual environment using immersive hardware. The content component hosts exercises and a monitoring system. These exercises include the virtual setting, the rules and objectives of the tasks, instructions, and interactions specific to the exercise. The user's point of view, in the immersive environment, is split into a secondary screen to allow the clinician to monitor. The last component concerns data management, where streaming data is used in a feedback loop to adapt the progression of exercises according to user performance, and detailed data such as movements, keystrokes, signals, Patient events, and performance are recorded for later analysis. (Gagnon Shaigetz et al.,2021)

3.2.6 Building safety training in real-time using XR cross-platform

The study of extended reality (XR) in the construction industry has been remarkably extensive, particularly in the field of safety training. The XR technology can create virtual simulations of hazardous construction sites, allowing workers to familiarize themselves with potentially hazardous environments before setting foot on a real construction site. This research describes a real-time communication protocol and data storage system tailored to an XR platform designed for real-time construction security training. The document focuses on two critical aspects: protocols for real-time communication and methods for storing data (Bao et al., 2022).

Cloud-Based Storage and Flexible Database Management

Enhancing previous methods that used fixed scenario frameworks, this research introduces a dynamic and adaptable framework. It allows users to upload files, to instantly create and edit scenario templates. It meets the need for an easy-to-use safety training system that maintains the interest of workers by eliminating the need for manual loading of scenarios. Instead, it leverages the Hypertext Transfer Protocol (HTTP) for flexible storage and retrieval of scenario models. The platform uses cloud-based storage because of the huge dimension of Industry Foundation Classes (IFC) files, text-based exports from Building Information Modelling (BIM) software. This solution handles not only IFC files but also three-dimensional objects from different sources, facilitating seamless creation and expansion of detailed scenario models.

Historically, databases were used to store IFC files and three-dimensional object files, usually relying on SQL with fixed relationships. However, the rigidity of this solution clashes with the need for flexibility of the new framework to support dynamically changing scenario models. Hence, a NoSQL (non-relational) database is preferred because of its horizontal scalability and the use of JavaScript Object Notation (JSON) for data storage. This format is ideal for the web and effective in storing model-specific information and user interactions in virtual environments. (Bao et al., 2022)

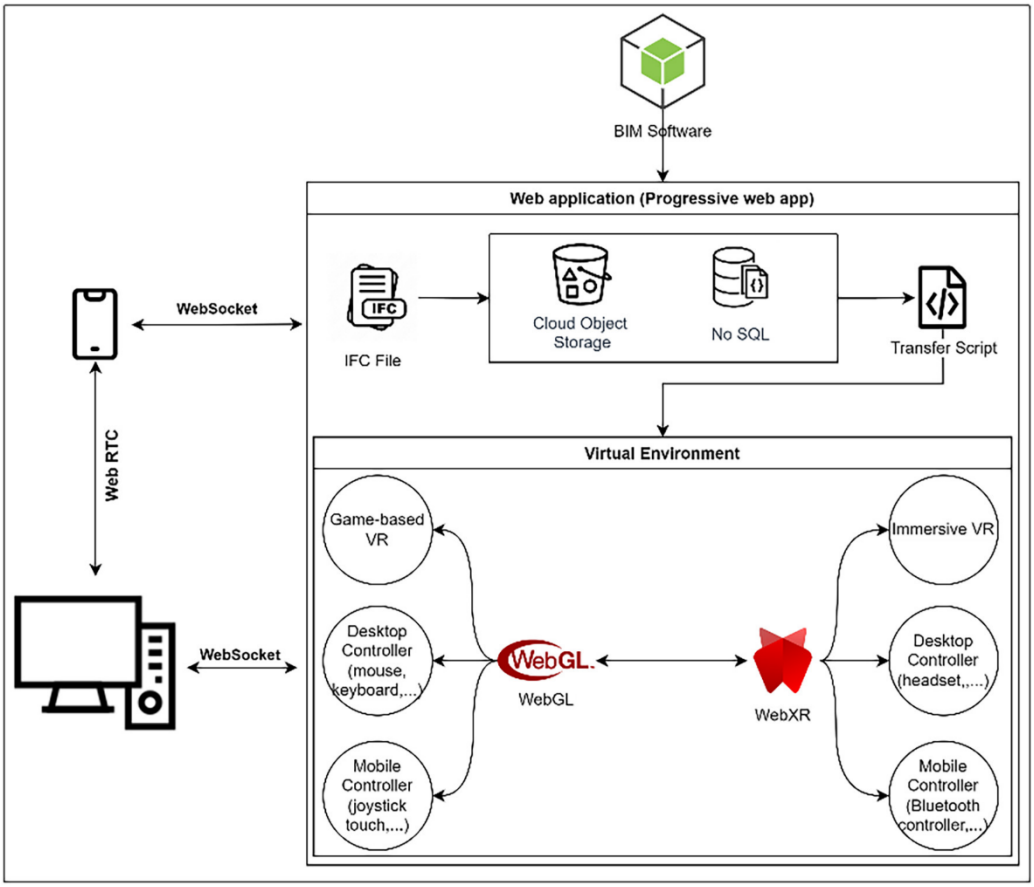

Figure 11: Comparison between game engine and web application in developing VR content from BIM. Bao et al. (2022)

Real-time Communication and Protocols

As mentioned before, in this architecture one of the communication models is the client-server one, but it can be inefficient for dynamic frameworks that require frequent updates. They proposed a framework that eliminates the need for manual updates or update requests allowing users to automatically receive and share new content. The platform uses web socket protocol to overcome the limitations of the traditional request-response model. As Bao et al. (2022) described, “It establishes a full-duplex, low-latency communication channel directly between the server and the user’s browser. This innovation facilitates the creation of virtual training spaces, identified as channels,

where users can join or leave at will, promoting both private and public interaction spaces".

Furthermore, WebRTC (a protocol that provides real-time communication (RTC) via simple application programming interfaces) enhances the direct communications of voice and video using peer-to-peer (P2P) architectures. It is efficient and improves the user experience with real-time data sharing (e.g., the user and objects positions). All these features play a key role in creating an immersive experience and are proof of the innovative approach to real-time communication in a virtual environment. This capability not only ensures efficient voice and video communication but also enriches the virtual environment with real-time data sharing, such as user location changes and interactions with virtual objects. These features are key to creating an immersive and interactive virtual training experience, highlighting the framework's innovative approach to real-time communication in virtual spaces. (Bao et al., 2022)

3.2.7 ImmersiveDeck: informal learning and first responder training XR platform

With ImmersiveDeck they have implemented a large-scale and low-cost multi-user XR hard- and software platform.

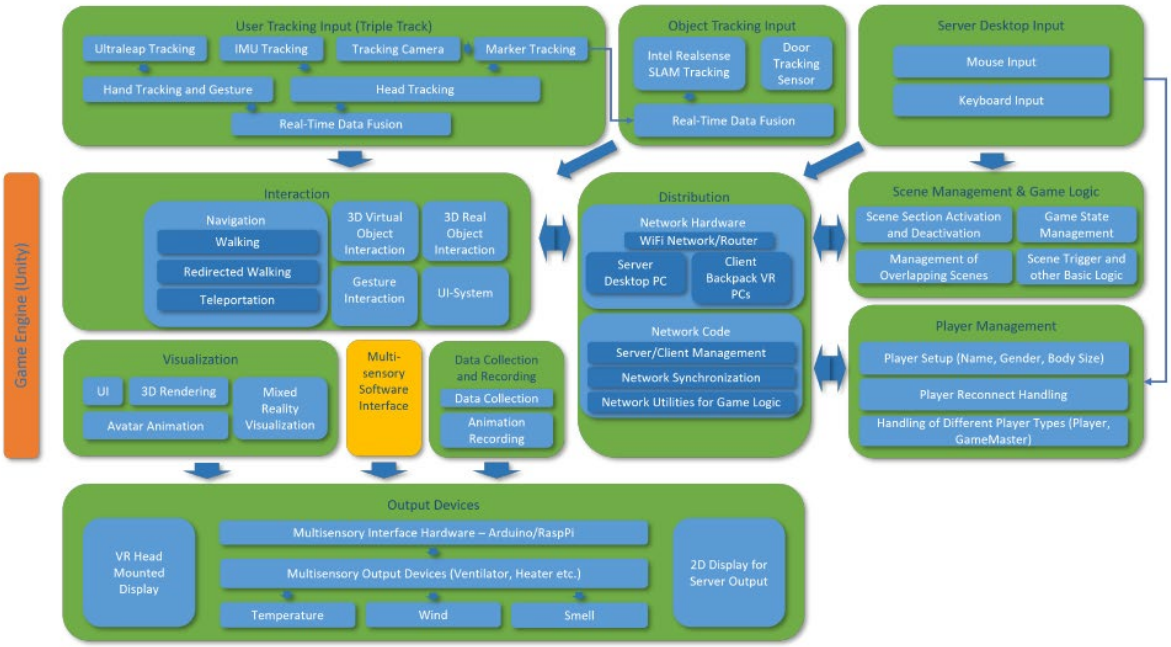

Figure 12: Overview of the ImmersiveDeck platform design and data flow. Schönauer et al. (2023)

Figure 12 shows the design and data flow of the ImmersiveDeck. The ImmersiveDeck includes positional tracking, even from room to room, for up to 10 people in which each user interacts using a Head Mounted Display (HMD). They choose the HP Reverb G2 or Oculus Rift CV1 which is connected to a VR backpack PC (HP Z VR backpack G2) and rendering of the virtual world is performed locally in the HMD on each user to minimize

latency. Adopting this setup, gives the possibility to make free-roam navigation, by adding absolute optical head tracking using markers, and natural hand interaction. (Schönauer et al., 2023)

The platform is strongly based on the game engine and Unity editor, while the software for tracking users and objects is encapsulated as a separate process and assembly implemented in C++ and C#. The ImmersiveDeck has the advantage of being flexible and scalable according to the number of users. The focus of the application is mainly on informal learning and training for rescuers. However, the platform is also adaptable for other uses, in fact, one of the strengths is flexibility. As Schönauer et al. (2023) said "the ImmersiveDeck facilitates the easy creation and integration of content, providing an encapsulated API for anyone to incorporate their content into the platform."

Schönauer et al. (2023) have identified 2 main goals:

- "Lower significantly the entry threshold for content providers to develop content for a multi-user multi-sensory XR platform, i.e., they should not need to be large development companies with dozens of programmers and designers to produce content for such a platform".
- "Democratisation of XR, a scalable solution for which everybody can freely and openly produce content".

The platform supports user avatar management like the creation using scripts or commands implemented in the Unity UI and customization by the administrator from the server UI. Moreover, the platform provides tools for scene management and game state management, helping temporal and spatial organization of the experience. Client-server architecture is used to implement a distribution in a multi-user setting, where a wireless router (Asus RT-AX88) is connected to the server, while the VR backpacks use their internal Wi-Fi cards. (Schönauer et al., 2023)

In this paper they present a challenge in a multi-user XR system with the goal of distributing large amount of tracking data but with the requirement of maintaining the latency as low as possible. In fact, as Schönauer et al. (2023) said "in situations when users touch each other or a virtual-real object, due to this trade-off, we configured the synchronization of object properties carefully. Unity Unet networking is used in the presented ImmersiveDeck implementation." But to enhance user convenience they have built a custom networking module for easy configuration of the network, creation of distributed objects, triggering of network callbacks, and automatically distributed scene management (Schönauer et al., 2023).

3.3 GAP Analysis for SUN expectations

Each of the works presented above introduces innovative systems and platforms covering various aspects of extended reality (XR). These systems found their scope in

different domains such as occupational safety, collaborative platforms, medical learning, cognitive rehabilitation, and construction safety training.

This paragraph presents a gap analysis focusing on identifying unmet needs, technological limitations, and opportunities for innovation that existing solutions have not fully addressed and that the SUN Platform can potentially tackle. Relevant gaps and areas that require further research and development based on the state of the art presented before are reported below.

Interoperability and Standardization

Frameworks and platforms (like MRTK, ARCore, Vuforia SDK, etc...) used for developing XR applications have been optimized for specific platforms and operating systems, so most of them support only a limited number of devices in an official way. This point makes visible the requirement of having interoperability to have users interact smoothly across devices and platforms. Finally, the state-of-the-art analysis highlights the requirement of having data and data format interoperability as well as communication protocols between applications and services, this translates into easy development of robust XR platforms.

Scalability for Large Multi-User Environments

The necessity of scalability emerges from discussing multi-user platforms and environments. This kind of scenario implies managing numerous users interacting with each other in a common space requiring significant challenges in terms of computational power and networking management. All these requirements can be translated into the necessity of a scalable system, capable of hosting an increasing number of users and preserving their performances and user experience.

Real-Time Data Integration and Collaboration

Integrating and collaborating data in real-time represents a key point of the XR technology. Systems should be capable of sharing and manipulating data synchronously as much as possible like the interactions in the same room. When a user interacts and modifies an object this should be reflected over all the other users sharing the same environment. This requires fast networks and efficient protocols for minimizing (as much as possible) latency and packet loss.

Security and Privacy

In the context of real-time user collaboration, the security and privacy of data play a fundamental role, given the immersive nature of the platforms and the high degree of interactivity involved. For instance, acquiring and managing sensitive data such as biometric and behavioural data is a delicate process and demands the maximum attention. XR platforms require more work and research on the implementation of strong security measures to protect exchanged data and guarantee access only for users that have been authorized.

4 SUN Architecture Design

4.1 Methodology

In order to design the SUN Reference Architecture a hybrid approach was followed. It includes a Bottom-Up approach, in which the scenarios, requirements, and specifications from the SUN Pilots were collected and analysed, and a Top-Down approach in which a detailed analysis of technical components was conducted.

The result of these analysis activities brings to the definition of the SUN Reference Architecture as a scalable, flexible, and configurable framework for implementing XR solutions in a secure and easy way.

The methodology adopted for the definition of the SUN Architecture and Technical Specifications can be represented in 4 main steps and it is shown in Figure 13.

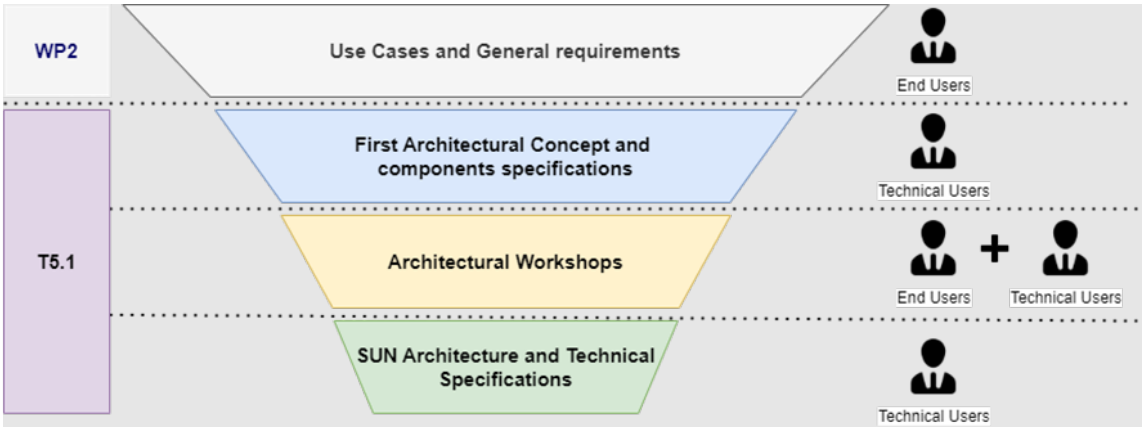

Figure 13: SUN Architecture Design Methodology

First, T5.1 aligned the expectations and the requirements elicited in WP2 in terms of pilots and scenarios, in order to define the perimeter for the design and the definition of the architecture.

As a second main activity, a collaboration with technical partners allowed us to collect technical component specifications (from WP3, WP4, and WP5), and starting from that a first concept of architecture was drafted.

The third step was to implement a series of architectural workshops that involved pilot and technical partners and allowed to refine the SUN Architecture iteratively and collaboratively.

The main outcome of this methodology were the SUN Architecture and the Technical Specifications reported in Section 5 and Section 6.

4.2 Pilots and Requirements Analysis

Starting already from the beginning of the project, two key tasks in WP2 have dealt with the definition of scenarios that will later be piloted in WP6 and the elicitation of user requirements for these scenarios. While the final public deliverable describing the final version of the user requirements and scenarios is due at month 22 in the project, intermediate versions were made available internally to the SUN consortium first at month 12 and later at month 16.

A co-creative methodology was used to engage the diverse stakeholders and define in more detail the 3 usage scenarios of SUN: physical rehabilitation, Industry 5.0, and cerebral rehabilitation. To facilitate the scenario definition process each pilot created a storyboard in a storytelling approach to give a clear perspective of a human-centric application of SUN in the specific framework. Next, each scenario presents an overview of the context, situation, and results as experienced now and as envisioned after SUN. The actors and stakeholders for each scenario are identified alongside the involved SUN technologies. User personas were then identified through interactive workshops for each scenario, presenting in a human-centric way the profile, needs, goals, pain points, and technology proficiency and behaviour.

Finally, based on all the previously gathered information from users and key stakeholders of each scenario, an initial list of user requirements was extracted. These were later grouped based on category and further analysed and revised in consortium-wide workshops. Through interactive workshops involving all stakeholders from the project, the requirements were prioritised using the MoSCoW methodology.

While the internal deliverable D2.2ii contains much more information about the scenarios and user requirements, we provide for the completeness of this document a summary in the following sub-sections. Moreover, “D6.1 SUN pilot planning” contains detailed information about the plans for implementing the scenarios in pilots.

4.2.1 Scenarios Pilot 1 - Physical Rehabilitation

Pilot 1 focuses on two scenarios: case 1 - rehabilitation of upper limb, and case 2 – rehabilitation of lower limb.

Clinical rehabilitation, often hindered by repetitive motor tasks and lack of continuous performance measurements, benefits from the integration of an XR system used alongside conventional therapy. This system, utilized in a hospital setting with both therapist and patient present, supports upper limb orthopaedic rehabilitation with tasks set within a serious game that simulates 3D construction of a virtual structure, enhancing engagement and measuring performance more precisely.

The XR environment not only visualizes correct movements but also identifies errors, such as abnormal muscle activation and compensatory movements, providing haptic feedback on these inaccuracies. It records detailed movement and postural data for ongoing analysis, offering adaptability for a variety of exercises based on patient

progress. This setup maintains direct interaction between the therapist and patient, improving communication and the overall effectiveness of the rehabilitation process.

The lower limb rehabilitation scenario is based on the use of a digital tool employing XR to assist and monitor individual motor learning in the context of a supervised personalized remote exercise rehabilitation program for the management of injuries/pathologies. It focuses on people affected by motor impairment due to orthopaedic illness, who require extended exercising for longer periods completed in in-patients, out-patients, and home environments.

Some of the main goals are to increase compliance with the rehabilitation protocol, increase patient engagement, and monitor in real-time physiological conditions providing immediate feedback to the patient and the therapist. Real-time monitoring will be through camera and augmented visual feedback, IMU, and s-EMG sensors acquired from wearables. Therapists, by exploiting XR (avatars) and AI algorithms outputs, will suggest modifications of the technique and type of exercise. Gamification of the exercises could be attempted using selective haptics to enhance the patient experience and make it more interesting.

The extracted user requirements for this pilot are presented in Table 1. Please note that this list represents the view on the user requirements at month 16 (when the internal version of D2.2 was released) and as the activities of the project continue this list might be revised. For the final description of the scenarios and the user requirements the final version of deliverable D2.2, to be released at month 22, should be consulted.

Table 1: Pilot 1 User Requirements

Pilot 1			
#	**User Requirement short description**	**Category[1]**	**Priority[2]**
UR1.1	Be informed of data being collected and processed	Legal	M
UR1.2	Legal basis in case the patient is injured using the SUN technology. (Especially if something happens outside the hospital have a clear understanding and agreement on who is responsible for that)	Legal	M
UR1.3	Receive periodic assessment of rehabilitation progress (range of motion achieved)	Exp	S

[1] User requirements are organised based on their category: legal, experiential (Exp), functional (Func)
[2] Each user requirement is assigned a priority based on the MoSCoW methodology: must have (M), should have (S), could have (C), won't have at this time (W)

UR1.4	Support the motivation of patients through several ways: Gamification of exercises / Engaging exercises / Performance as a motivator	Exp	S
UR1.5	User friendly / intuitive interface for patients but advanced enough for physiotherapist to measure the progress	Exp	S
UR1.6	Have instructions and feedback system while performing the exercise	Exp	M
UR1.7	Have customized exercises to mimic the extent of motion the patient uses during their regular sport activity	Exp	C
UR1.8	Have a clear mapping of the rehabilitation journey	Exp	C
UR1.9	Be able to perform the physiotherapy exercises without the supervision of the physiotherapist (not from the beginning but after a few sessions done together with the rehabilitation expert). There should be good guidance (for e.g. using an avatar in XR)	Exp	C
UR1.10	Simplified way of seeing the EMG findings	Exp	C
UR1.11	Ensure minimal technical issues even when using different sensors, esp. for elderly patients	Exp	S
UR1.12	Have short term / immediate feedback during the rehabilitation process to keep the patient motivated	Func	M
UR1.13	Visual feedback (e.g. visual representation of the exercises, avatar) on the performance of the rehabilitation exercises (based on a set of metrics like range of motion).	Func	M
UR1.14	Measure patients progress with basic metrics recorded during the exercises (i.e. completion time, knee or shoulder angle) or	Func	M

	Measure patients progress with more advanced data aggregation / AI algorithms mixing several metrics in the longer timespan of the therapy.	Func	W
UR1.15	The physiotherapist needs one (or more) fine-tuned XR exercises according to the patient's performance.	Func	C
UR1.16	Physiotherapist needs to be able to record the clinical results of their patients using objective parameter recording (i.e. record raw data from the session).	Func	S
UR1.17	Access additional data (e.g. breath, emg/heart rate, etc) and information regarding the outcome of the patient progress achieved (e.g. the range of motion)	Func	C
UR1.18	To provide exercises able to improve patient's motor performance	Func	M
UR1.19	Have an avatar showing how the exercise is performed correctly	Func	M
UR1.20	Have a simplified system for gait analysis (kinematic parameters of the lower limb) and compare the lower affected limb to the healthy one. Need to measure gait for a curved trajectory (not only straight line).	Func	S
UR1.21	System enables the psychotherapist to monitor the patient's emotional state	Func	S
UR1.22	Avatar is able to move normally - as if the patient was already recovered	Func	M
UR1.23	[For upper limb exercises] The patients are able to see the ideal movement to reach the object and try to follow this.	Func	S

UR1.24	Therapist can access the same XR environment as the patient (i.e via an HMD) to provide guidance and observe.	Func	C

4.2.2 Scenarios Pilot 2 – Industry 5.0

Pilot 2 focuses on Industry 5.0 and envisions also two scenarios: one related to Personal Protective Equipment (PPE) training and practice, and one related to shop floor safety & object tracking.

In the first case, workers navigate the hazards of handling massive metal bars on a bustling shop floor. Despite their efforts, the potential for accidents and errors due to human limitations and distractions remains high. To minimize these risks, every new employee undergoes thorough safety training. Additionally, the company emphasizes a culture of collaboration and accountability from the onset to further safeguard the workplace. AR training helps with the onboarding process and allows simple (re-)training of any employee, therefore ensuring an ongoing adherence to the regulations. The training is split into 2 phases.

Phase 1 of the training involves the worker receiving instruction on proper PPE usage in a dedicated room, using an AR application that provides step-by-step visual guidance on how to correctly wear the necessary safety gear for specific production tasks. The worker actively participates by putting on the PPE as instructed, and upon completion, the trainer verifies the correctness, paving the way for the next phase.

In Phase 2, the worker practices his newly acquired knowledge on the shop floor by analysing the PPE usage of his colleagues. Using camera equipment and the HoloLens 2, an object detection algorithm helps identify correct or incorrect PPE usage by displaying bounding boxes and highlighting any discrepancies. He is tasked with advising colleagues on corrective measures for any observed PPE violations. After a set period, the worker reports back to his trainer with his findings, such as compliance rates among colleagues, without recording any personal data.

The second scenario case is also focusing on floor employee safety and optimisation of tasks. In the shop floor of a manufacturing company, it is common for the transit corridors to often have objects obstructing the passage, among various things out of place. Workers will wear HoloLens 2 and the system should address employee safety by utilising external cameras, as well as the HoloLens 2 cameras, object recognition, and risk prevention through removal. The second scenario aims to generate warnings on wrongly positioned material. Additionally, since containers must be moved due to raw material delivery and waste retrieval, a solution is proposed to support the workers in coordinating the container's movement. The scenario provides a tool to understand that the containers must be replaced with new raw material or remove full waste containers.

This scenario involves task optimization: generating a list of tasks based on container statuses. Tasks include moving containers to the correct position, emptying waste containers, and refilling raw material containers. This system enhances safety and reduces stress by optimising tasks and ensuring proper container management. Ultimately, SUN project outcomes create a safer and more efficient shop floor environment by utilising technology to address safety concerns and streamline task coordination.

The extracted user requirements for this pilot are presented in Table 2. Please note that this list represents the view on the user requirements at month 16 (when the internal version of D2.2 was released) and as the activities of the project continue this list might be revised; for the final description of the scenarios and the user requirements the final version of deliverable D2.2, to be released at month 22, should be consulted.

Table 2: Pilot 2 User Requirements

Pilot 2			
#	**Requirement short description**	**Category[3]**	**Priority[4]**
UR2.1	Inform about data being collected and processed.	Legal	M
UR2.2	The SUN application does not track personal user data.	Legal	M
UR2.3	The SUN application supports different interaction and input methods.	Exp	M
UR2.4	The users (workers) are informed about the purpose of the PPE instructions.	Exp	M
UR2.5	The SUN application for PPE training and practice is utilised to train the user (worker) in appropriate PPE usage.	Exp	M
UR2.6	Automated detection if/when the PPE is not worn/used correctly, through object recognition functionality of the SUN system.	Exp	M

[3] User requirements are organised based on their category: legal, experiential (Exp), functional (Func)
[4] Each user requirement is assigned a priority based on the MoSCoW methodology: must have (M), should have (S), could have (C), won't have at this time (W)

UR2.7	Technology comfortable to wear and not invasive. Not intrusive to the tasks that need to be done.	Exp	C
UR2.8	The application has to be user friendly to allow users of all backgrounds to benefit from the training.	Exp	S
UR2.9	The training is conducted without additional guidance from a trainer.	Exp	C
UR2.10	The training is visualised through the most useful medium. This can be an avatar, text and/or images, videos, etc.	Tech	M
UR2.11	The trainer can switch between training mode “Training” and “Practice” through the system.	Tech	C
UR2.12	The SUN technology recognises the different PPE worn by staff members visible through the SUN camera system (external cameras and/or HoloLens 2)	Func	M
UR2.13	The SUN technology recognises if PPE is missing.	Func	S
UR2.14	The SUN technology recognises if the wrong PPE is worn.	Func	C
UR2.15	The application tracks event messages.	Func	S
UR2.16	Visual indicators or highlights are placed around relevant recognised PPE.	Func	S
UR2.17	Visual indicators are colour coded, based on system verification results.	Func	S
UR2.18	Training summary of “Practice” part generated as pdf, allowing the trainer and trainee to discuss results and performance.	Func	S
UR2.19	The safety application has a minimal design concept. Only the most relevant information can be displayed, to avoid obstructing the view of the user.	Func	S
UR2.20	The application has a “Task Confirmed” functionality to set a task as complete.	Func	M

UR2.21	The event "Task Confirmed" is transmitted automatically to the backend system by the application, to avoid lengthy interaction of the user.	Func	M
UR2.22	The SUN technology recognizes the status of various objects, e.g., container = empty; container = full.	Func	M
UR2.23	The SUN technology makes decisions based on object status, e.g., container = empty, decision = refill.	Func	S
UR2.24	SUN technology contains an algorithm to set priority of a task.	Func	S
UR2.25	The SUN technology triggers a task-update event, in case a higher priority task appears and communicates it to the application.	Func	S
UR2.26	The application displays task related information in AR, e.g., new task, task-update, additional warning information, etc.	Func	S
UR2.27	The user (worker) is able to manually change the priority of the task.	Func	C
UR2.28	[To enable UR2.27] UI elements are available to enable the user to conveniently adjust the priority of the task.	Func	C

4.2.3 Scenarios Pilot 3 – Cerebral rehabilitation

Pilot 3 is also focusing on two different cases: patients who are recovering from stroke or spinal cord injury and patients who are suffering from Apathy.

Stroke and incomplete tetraplegic patients are usually confined by physical limitations. This is why VR and EMG sensors can help them "go" (virtually) to known places, potentially meeting their family and friends (who are also using VR). The virtual environment is well enough crafted so that it seems "real". The platform translates the residual EMG activity from the patient's remaining arm muscles into nuanced hand and wrist movements within the virtual world, enabling them to explore as if they were physically there. Virtually navigating through familiar scenes, Martin encounters avatars

of his real-life friends and family. Enhancing the immersion, thermal feedback replicates the sensation of touch. While the patient remains in his clinic bed, the multisensory virtual experience helps him make the time pass faster. It allows him to communicate with his loved ones in places he really enjoys, providing a temporary respite from the psychological burden of his physical restrictions.

In the case of apathy patients, the goal is to use the SUN platform to conduct motivational therapy which has the goal to favour neuronal plasticity in the reward system by combining an immersive, pleasant experience with brain stimulation (delivered at the hospital). The patient wears the VR/AR headset and task instructions are provided; she has to perform a series of hand and wrist movements to get a reward. Within this motivation task, a plasticity-inducing brain stimulation protocol is delivered. The motivation task pre-activates the residual function of the reward network of the brain, which can be further supported by this.

The extracted user requirements for this pilot are presented in Table 3. Please note that this list represents the view on the user requirements at month 16 (when the internal version of D2.2 was released) and as the activities of the project continue this list might be revised; for the final description of the scenarios and the user requirements the final version of deliverable D2.2, to be released at month 22, should be consulted.

Table 3: Pilot 3 User Requirements

Pilot 3			
#	**Requirement short description**	**Category[5]**	**Priority[6]**
UR3.1	User-friendly devices (easy to wear and to use)	Tech	C
UR3.2	Legal consent from the patients	Legal	M
UR3.3	Data being collected and processed - surveillance.	Legal	M
UR3.4	Interact (e.g., grasp a cup/mug, receive a caress) and communicate (virtual keyboard or microphone) with other people in VR	Exp	M

[5] User requirements are organised based on their category: legal, experiential (Exp), functional (Func)
[6] Each user requirement is assigned a priority based on the MoSCoW methodology: must have (M), should have (S), could have (C), won't have at this time (W)

UR3.5	Users can perform (alone) in VR activities they can do in real life (e.g., read a book, play a game…)	Exp	C
UR3.6	Users can perform activities in VR together with other users (e.g., other family members)	Exp	S
UR3.7	Receive immersive experience (touch and feel) to an action the patient performs	Exp	M/S
UR3.8	Keep the patient engaged and motivated (engaging technology / interface, VR)	Exp	S
UR3.9	User friendly application (is intuitive to use by the user without someone's constant help)	Exp	C
UR3.10	The system enables the therapist to monitor the patient's emotional state (also as a way to understand the benefit of using VR)	Exp	C
UR3.11	The user is allowed to interact with several objects from which it is possible to choose from / freedom to interact with different options	Func	M
UR3.12	Ability of giving vocal commands or talk with another user	Func	C
UR3.13	Reconstruct physical environments or objects credibly	Func	~~C~~
	Reconstruct physical environments or objects credibly	Func	C
UR3.14	User is able to use the system and communicate with other users also when not being able to speak	Func	C
UR3.15	Other users (e.g., relatives of the patient) can access the virtual environment without having a physical impairment using standard controllers	Func	C
UR3.17	Users of the VR have the possibility to navigate the environment	Func	M

4.3 Components specifications

In parallel with the pilots and requirements analysis, also a technical evaluation of the SUN components was done.

It was important to identify, among the SUN technical components, specific information to be analysed and taken into account for the design of the SUN Reference Architecture.

In fact, adopting a top-down approach that focuses on the system components planned for implementation adds significant value to the design of the SUN Architecture. This strategy leverages the deep knowledge and experience of the technical partners involved, allowing for the anticipation and addressing of each component's technical needs at an early stage of the process. Table 4 summarizes the list of components considered during the analysis and related partner involvement. This final list was consolidated after the three rounds of collaborative workshops described in Section 4.4

Table 4: SUN Components List

Component	Task	Responsible
AR App Pilot 1 (Collaboration, Glanceable Interfaces)	T4.1	TUC
EMG decoding system for hand and wrist kinematics	T3.2	EPFL
Wearable system for thermal feedback	T3.1	EPFL
High-performance Interactive Streaming	T4.3	HOLO
Haptic device: Distributed Wearable Haptic system for directional hints and cues	T3.1	SSSA
Haptic device: Wearable Haptic system for manipulation cues	T3.1	SSSA
Postural Assessment	T3.2	TG
Wearable-based pose estimation	T3.2	TG
Energy efficient PPG-based heart rate estimation	T3.2	TG
AI Reconstruction of 3D Models	T4.4	CNR
3D reconstruction using NERF	T4.4	CERTH
IGOODI End2End Avatar Production Pipeline	T3.4	IG
Fused Conditional Denoising Diffusion Probabilistic Model	T4.5	CNR

Pose Estimation	T3.2	CERTH
Gesture Recognition	T3.3	CERTH
Emotion Recognition	T3.3	CERTH
Multimodal fusion module	T3.3	CERTH
Logistics Processes Optimization/Supporting decision making	T6.2	UPV
Tokenized Platform	T5.2	ENG
Cyber Threat Detection	T5.3	UoG
Non-invasive brain stimulation	T6.2	EPFL
Open-Vocabulary Object Detection (aka Semantic Recognition for movable objects)	T3.5	CNR
Semantic 3D Scene Reconstruction	T3.5	CNR
Acquisition of physical properties for 3D objects	T4.2	CNR
VR APP for pilot 3	T6.2	ENG
Holo AR App - Pilot 2	T6.2	HOLO
Room Scan App	T6.2	HOLO

To collect the necessary information, a survey among technical partners was conducted using the template available in Appendix A, for each of the components listed in Table 4.

Table 5: SUN Components Description - Wearable system for thermal feedback

WP and Task reference	WP3: Novel human-machine interaction and contextualisation T3.1 Haptic interfaces
Responsible	EPFL, Jonathan Muheim (jonathan.muheim@epfl.ch)
Name of Component/Service:	Wearable system for thermal feedback

Type	Hardware
Functionality	This system is composed of two main modules: up to two thermal displays and a control unit. The thermal displays are worn on the forearm (and/or the hand, TBD). It heats up/cools down the skin in the human non-painful range (15-42 °C). The control unit regulates the power input to the thermal display to attain a desired setpoint (PID controller). The system can be used under three different working modes: – A constant temperature is applied to the skin. – The controller tracks a moving setpoint in real-time. – A predefined thermal signature (eg: the illusion of contact with a material) is played. Additionally, the system keeps track of the setpoint and temperature of the thermal display.
Input Connections & Interfaces	Touch events and virtual temperature are received from the VR app through TCP.
Output Connections & Interfaces	The temperature change is applied on the skin of the user
Software Requirements/Development Language	Microcontroller firmware: C, C++ GUI: python
Hardware Requirements	The system consists of a Peltier module assembled on a passive heatsink. A temperature sensor is mounted at the interface with the skin. An embedded microcontroller is in charge of implementing the closed-loop control and acquiring and sending the data (timestamp, new setpoint, temperature of the display).

Status of the development of the component	Partially developed. The working principle has been validated with prototypes. Some improvements in the hardware are necessary.

Table 6: : SUN Components Description - Haptic device: Distributed Wearable Haptic system for directional hints and cues

WP and Task reference	WP3 – T3.1
Responsible	Cristian Camardella /Daniele Leonardis
Name of Component/Service:	Haptic device: Distributed Wearable Haptic system for directional hints and cues
Type	Hardware
Functionality	The wearable haptic system is able to provide skin stretch and pressure feedback in a “differential” way to achieve directional hint on multiple Degree of Freedom (DoF). Directional hints are used to better perform reaching tasks in virtual/augmented reality environments
Input Connections & Interfaces	Directional hints are performed based on the “distance” from the target (e.g., a certain rotation value on a single DoF). Thus, the input interface concerns a “setpoint” on which the feedback is built. These inputs can be provided through UDP packets over WiFi network
Output Connections & Interfaces	None (haptic devices provide skin stretch to the user as the final output)
Software Requirements/Development Language	The software requirements concern the (*potential*) application interface with which external software can communicate. The haptic device is controlled using a WIFI-ready microcontroller. The control scheme is written

	in C-like. The *potential* application interface will be written as a C# component.
Hardware Requirements	There are no specific hardware requirements. A WiFi connection is needed on the host PC, to communicate with the device. Potential future wearability issues can be evaluated after testing the device in an operative environment
Status of the development of the component	The prototype is partially developed and tested in the laboratory (TRL4)

Table 7: SUN Components Description - Haptic device: Wearable Haptic system for manipulation cues

WP and Task reference	WP3 – T3.1
Responsible	Cristian Camardella / Daniele Leonardis
Name of Component/Service:	Haptic device: Wearable Haptic system for manipulation cues
Type	Hardware
Functionality	This wearable haptic system, in the shape of a glove, will deliver contact cues related to simulated virtual manipulation, or augmented cues in augmented reality interaction with real objects.
Input Connections & Interfaces	Haptic rendering and cues are generated by the VR/XR environment in real time. Input data can be both discrete events associated with discrete haptic cues (i.e. contact event, or other alerts) or real-time signals generated by the physical engine in the VR (i.e. indentation and relative velocity) between virtual finger and object surface, or relative velocity between virtual finger and objects. In the

	latter, the final rendered signal is generated in real time at a high sample rate on the embedded electronics of the device. The above inputs can be provided through UDP packets over WiFi network
Output Connections & Interfaces	None (haptic devices provide haptic cues to the user as the final output)
Software Requirements/Development Language	The software requirements concern the (*potential*) application interface with which external software can communicate. The haptic device is controlled using a WiFi-ready microcontroller. The control scheme is written in C-like. The *potential* application interface will be written as a C# component. In the real-time rendering mode, a sample rate of about 100 Hz for the UDP messages is required
Hardware Requirements	There are no specific hardware requirements. A WiFi connection is needed on the host PC, to communicate with the device. Potential future wearability issues can be evaluated after testing the device in an operative environment
Status of the development of the component	The prototype is partially developed and tested in the laboratory (TRL4)

Table 8: Sun Components Description - Postural Assessment

WP and Task reference	WP3 - T3.2
Responsible	ThinGenious (TG) Panagiotis Kasnesis (pkasnesis@thingenious.io)

Name of Component/Service:	Postural Assessment
Type	Software, but hardware parts like sensors are included.
Functionality	This machine learning-based application will process EMG and IMU signals (provided by Delsys sensors) to decide whether the patients perform correctly the lower limb rehabilitation exercises (seated leg extensions and squats). Start data recording from Delsys SDK server and after pairing the necessary sensors with it, the following software functions take place as pipeline. 1. Data streaming in python 2. Data preprocessing in python 3. Data processing using ML in python
Input Connections & Interfaces	• **Data streaming:** Trigno Research+ sensors (+ 3rd party sensors integrated in the Trigno Research+ System). Through RF are sent to Trigno Research+, which are sent afterward to a Windows PC using a serial connection (i.e., usb cable). • **Data preprocessing:** has raw bytes as input from the data streaming interface, and converts them to float, and splits them to different sensor modalities. Afterward, data normalization (e.g. z-score) takes place. • **Data processing**: has input coming from the preprocessing module. There could be 2 different cases here:

	• The ML part runs on the same device with the preprocessing. It has as input numpy arrays. • The ML part runs on the SUN platform over the web. In this case will be dockerized having as input interface a gRPC/RESTful API using JSON or Protobuf format.
Output Connections & Interfaces	• **Data streaming:** Delsys API or SDK could be used here to get the output raw bytes. • **Data preprocessing:** Output will be preprocessed data. API could be developed for getting the data however it is not considered optimal to transfer over web or local network these amounts of data. Data could be sent in JSON/ Protobuf format to other SUN's modules. • **Data processing**: The output is processed data, i.e., the machine learning model's inference. The output data will be in JSON and will be sent to the SUN platform using pub/sub protocol (e.g., MQTT messages).
Software Requirements/Development Language	PC equipped with Windows 10/11 (will be probably provided by TG) Python programming language, exploiting several numerical/deep learning frameworks (numpy, pytorch, etc). In case of being included in the core SUN platform, the provided modules will be dockerized to enable seamless integration.
Hardware Requirements	**Windows PC**

	One USB 2.0 port 2.0 GHz processor clock speed (minimum) 2 GB system memory (minimum) 1 GB hard disk storage (minimum) 50 GB hard disk storage
Status of the development of the component	The component is partially developed (ML part is missing) - TRL 4

Table 9: SUN Components Description - Wearable-based pose estimation

WP and Task reference	WP3 - T3.2
Responsible	ThinGenious (TG) Panagiotis Kasnesis (pkasnesis@thingenious.io)
Name of Component/Service:	Wearable-based pose estimation
Type	Software, but hardware parts like sensors are included.
Functionality	We will use XSENS IMU sensors provided by Movella to estimate the inverse kinematics of the patients. Start data recording from MT SDK[7] and after pairing the necessary sensors with it, the following software functions take place as pipeline. 1. Data streaming in python 2. Data preprocessing in python 3. Data processing using ML in python
Input Connections & Interfaces	• **Data streaming:** 6 or 7 Xsens MTw Awinda sensors produce 3-axial accelerometer, gyroscope,

[7] https://www.xsens.com/hubfs/Downloads/usermanual/MT_Manager_user_manual.pdf

	magnetometer signals. Through RF are send to Awinda Station (i.e., gateway), which are send afterward to Windows PC using serial connection (i.e., usb cable). • **Data preprocessing:** has raw bytes as input from the data streaming interface, and converts them to float, and splits them into different sensor modalities. Afterward, data normalization (e.g. z-score) takes place. • **Data processing**: has input coming from the preprocessing module. There could be 2 different cases here: • The ML part runs on the same device with the preprocessing. It has as input numpy arrays. • The ML part runs on the SUN platform over the web. In this case will be dockerized having as input interface a gRPC/RESTful API using JSON or Protobuf format.
Output Connections & Interfaces	• **Data streaming:** MT SDK to be used here to get the output raw bytes. • **Data preprocessing:** Output will be preprocessed data. API could be developed for getting the data however it is not considered optimal to transfer over web or local network these amounts of data. Data could be sent in JSON/ Protobuf format to other SUN's modules. • **Data processing**: The output is processed data, i.e., the machine learning model's inference, providing 3D values for 20 joint angles. The output data will be in JSON and will be sent to SUN platform using pub/sub protocol (e.g., MQTT messages).

Software Requirements/Development Language	PC equipped with Windows 10/11 (will be probably provided by TG) Python programming language, exploiting several numerical frameworks (e.g., Numpy, PyTorch). In case of being included in the core SUN platform, the provided modules will be dockerized to enable seamless integration.
Hardware Requirements	**Windows PC** One USB 2.0 port 2.0 GHz processor clock speed (minimum) 2 GB system memory (minimum) 1 GB hard disk storage (minimum) 50 GB hard disk storage
Status of the development of the component	The component is partially developed - TRL 4

Table 10: SUN Components Description - Energy efficient PPG-based heart rate estimation

WP and Task reference	WP3 - T3.2
Responsible	ThinGenious (TG) Panagiotis Kasnesis (pkasnesis@thingenious.io)
Name of Component/Service:	Energy-efficient PPG-based heart rate estimation
Type	Software, but hardware parts like sensors are included.
Functionality	We will use PPG, IMU, etc sensors provided by Empatica E4 to monitor the heart rate of the patients.

	Start data recording from the smartphone and after pairing it with E4, the following software functions take place as a pipeline. 1. Data streaming 2. Data preprocessing in Android 3. Data processing using ML in Android 4. Data postprocessing in Android
Input Connections & Interfaces	• **Data streaming:** captures and transmits data from the wristworn device are sent to the mobile phone through BLE. • **Data preprocessing:** has raw bytes as input from the data streaming interface, converts them to float, and splits them to different sensor modalities. Afterward, data normalization (e.g. z-score) takes place. • **Data processing**: has input coming from the preprocessing module. The ML part runs on the smartphone and has as input numerical arrays of size 4 x 256. • **Data postprocessing:** clips the output values in case the prediction is more or less 10% of the averaged 10 last estimated values.
Output Connections & Interfaces	• **Data streaming:** Empatica Android SDK to be used here to get the output raw bytes. • **Data preprocessing:** Output will be preprocessed data (numerical array of size 4x256), passed to data processing module. • **Data processing**: The output is processed data, i.e., the patient's heart rate estimated by the machine

	learning model's inference. The output data will be sent to the data postprocessing module. • **Data postprocessing**: The output is one float value sent in JSON format to SUN platform using pub/sub protocol (e.g., MQTT messages).
Software Requirements/Development Language	Android programming language, exploiting TensorflowLite.
Hardware Requirements	**Android smartphone** 1.0 GHz processor clock speed (minimum) 2 GB system memory (minimum) 8 GB hard disk storage (minimum)
Status of the development of the component	The component is partially developed - TRL 4

Table 11: SUN Components Description - Camera Based Pose Estimation

WP and Task reference	WP3, T3.2
Responsible	CERTH (Ilias Poulios, Spyridon Symeonidis)
Name of Component/Service:	Camera-Based Pose Estimation
Type	Software
Functionality	The Pose Estimation component aims to extract the 3D poses of people depicted in RGB videos to be used to identify their activities, create animations to be integrated into 3D avatars, and assess people's movements (for rehabilitation scenario) based on a given ground truth. A preliminary list of functions is the following: 1. Estimation of 2D poses

	2. Estimation of 3D poses 3. Motion transfer to 3D avatars 4. Movement assessment
Input Connections & Interfaces	This component will receive RGB input from the camera. Video streaming will be supported as input.
Output Connections & Interfaces	The outputs of this component will be: • 2D/3D body landmarks (e.g., JSON file) • Longitudinal bone rotation information A service will be created for communication with other technical components.
Software Requirements/Development Language	**Programming Language:** Python 3.8-3.11 **Tools/Libraries:** Anaconda, Keras, Tensorflow, PyTorch, OpenCV, FFMPEG, Mediapipe
Hardware Requirements	**Minimum system requirements:** Operating system: Windows 10 or 11 / Ubuntu 20.04 RAM: 16GB Free disk space >=50GB GPU-RAM >= 12GB (e.g., NVIDIA GeForce RTX 3090) **Camera's specifications:** Resolution: Full HD 2 megapixel Image sensor: 1/2.7" 2- megapixel progressive scan CMOS Video Resolution: 1920 x 1080 (16:9) Lens: 2.39 mm, F2.0, Fixed Indoor camera Connectivity: Wired or Wireless depending on use case Example: https://www.golmar.es/products/cip-21c2w#product
Status of the development of the component	The component will be developed from scratch for SUN project. Starting TRL: 3

Table 12: SUN Components Description - Gesture Recognition

WP and Task reference	WP3, T3.3
Responsible	CERTH (Ilias Poulios, Spyridon Symeonidis)
Name of Component/Service:	Gesture Recognition
Type	Software
Functionality	This component interprets and understands the gestures made by a person's hand or hands. A suitable deep-learning model will be used for the classification process that identifies the hand gesture. A preliminary list of functions is the following: 1. Receive camera input 2. Perform classification 3. Prepare output
Input Connections & Interfaces	The input of this component will be direct RGB video input, from external or headset camera, depending on each use case's needs.
Output Connections & Interfaces	The output will be the predicted gesture, along with a confidence level (e.g., JSON file). A service will be developed for the communication with other technical components. **Example gesture values:** "OK", "Like", "Dislike", "Stop"
Software Requirements/Development Language	**Programming Language:** Python >= 3.7 **Tools/Libraries:** Anaconda, Keras, Tensorflow, PyTorch, OpenCV, FFMPEG

Hardware Requirements	**Minimum system requirements:** Operating system: Windows 10 or 11 / Ubuntu 20.04 RAM: 16GB Free disk space >=50GB GPU-RAM >= 12GB (e.g., NVIDIA GeForce RTX 3090) **Camera's specifications (for external camera):** Same camera as Pose Estimation
Status of the development of the component	The component will be developed from scratch for SUN project. Starting TRL: 3

Table 13: SUN Components Description - Face Emotion Recognition

WP and Task reference	WP3, T3.3
Responsible	CERTH (Ilias Poulios, Spyridon Symeonidis)
Name of Component/Service:	Face Emotion Recognition
Type	Software
Functionality	This component aims to classify humans' emotions based on facial expressions. The algorithm will focus on the lower face features (e.g., lips) that are visible in the case of VR users. A preliminary list of function is the following: 1. Detect face landmarks 2. Prepare features 3. Emotion extraction based on deep learning classification model 4. Prepare output

Input Connections & Interfaces	This component will receive RGB input from camera. Video streaming will be supported as input.
Output Connections & Interfaces	The output will be the predicted emotion, along with a confidence level (e.g., JSON file). A service will be developed for the communication with other technical components. **Example emotion values:** "happy", "sad", "surprise", "fear", "disgust", "anger", "neutral"
Software Requirements/Development Language	**Programming Language:** Python >= 3.7 **Tools/Libraries:** Anaconda, Keras, Tensorflow, PyTorch, OpenCV, FFMPEG
Hardware Requirements	**Minimum system requirements:** Operating system: Windows 10 or 11/ Ubuntu 20.04 RAM: 16GB Free disk space >=50GB GPU-RAM >= 12GB (e.g., NVIDIA GeForce RTX 3090) **Camera's specifications:** Same camera as Pose Estimation
Status of the development of the component	The component will be developed from scratch for the SUN project. Starting TRL: 2

Table 14: SUN Components Description - Multimodal fusion module

WP and Task reference	WP3, T3.3
Responsible	CERTH (Vasileios-Rafail Xefteris, Spyridon Symeonidis)
Name of Component/Service:	Multimodal fusion module
Type	*Software*

Functionality	The module is responsible for the fusion of the outputs of each modality. Results received from the visual analysis and sensor analysis tools are processed by the multimodal fusion module in order to produce a unified outcome that improves the overall performance in 3D pose estimation and emotion recognition.
Input Connections & Interfaces	Sensor analysis and, visual analysis tool
Output Connections & Interfaces	Depends on which results will be fused. Output format will be similar to the unimodal components' one. A service will be developed for the communication with other technical components.
Software Requirements/Development Language	Python version 3.7 or later scikit-learn and tensorflow libraries
Hardware Requirements	A server with Windows 10 or 11 OS.
Status of the development of the component	The component will be developed from scratch. Starting TRL: 2

Table 15: SUN Components Description - IGOODI End2End Avatar Production Pipeline

WP and Task reference	WP3 T3.4
Responsible	IGOODI, Ioannis Paraskevopoulos <ioannis.p@igoodi.it>
Name of Component/Service:	**IGOODI End2End Avatar Production Pipeline**
Type	Software
Functionality	An end2end, scalable, mass production platform for hyper realistic avatars to populate en-mass XR, human centred applications, and use cases. The pipeline consists of a set

	of individual steps each responsible for a specific operation on Avatar production, starting off with the template fitting, to rigging, texture wrap and cleaning, clothing, Level of Detail definition and finally asset extraction in various formats. This asset can be deployed as a platform on a cloud to enable scalability.
Input Connections & Interfaces	**Production Pipeline:** The production pipeline receives as input a raw model scanned from various types of inputs and creates as an output a complete and usable Avatar. Interfacing with multimodal scan technologies (e.g., nerfs to 3d, rgbd scanners, laser scanners, photogrammetry). The production pipeline is an asset that is not directly connected to any other asset in the overall architecture but enables the overall solution with the scalable production of Avatars.
Output Connections & Interfaces	**Avatar into Engine:** As an output the Avatar Asset (3D model) will be used in the use cases for the pilots. As such it needs to be compatible with the engines of choice (e.g., Unity, Unreal other?). **Motion capture:** It also interfaces with the sensors capturing movement of the final users and it needs to be mapped on the Avatar. **Data:** it incorporates a priori data (anthropometric measurements) as well as accumulated data captured during the sessions (e.g., performace, behaviour, etc) to enable context awareness
Software Requirements/Development Language	**Development of Pipeline:** Python Pytorch

	(Versions TBC) **Avatar in use cases:** Compatible on demand with all major game/graphics engines (Unity, Unreal, webgl, three.js…). Need to confirm which one is the engine of choice
Hardware Requirements	N/A
Status of the development of the component	To be developed from scratch, currently under development TRL1à4/5

Table 16: SUN Components Description - Open-Vocabulary Object Detection

WP and Task reference	T3.5 - AI to learn objects for the virtual world
Responsible	CNR - Fabio Carrara
Name of Component/Service:	Open-Vocabulary Object Detection
Type	Software
Functionality	This component can detect objects in RGB images without relying on predefined categories. Instead, it allows users to specify the categories of interest through natural language descriptions.
Input Connections & Interfaces	- RGB(D) frames from the video stream, probably passed via API (theoretically direct input from sensors would be possible, but HW requirements hardly will fit in the possibilities of XR device). - textual data (categories to be detected) from the user via API
Output Connections & Interfaces	Outputs (e.g., bounding boxes) available via API Possible consumers: XR Apps / UI

Software Requirements/Development Language	OS or containers (Docker, podman) for running Python scripts with CUDA-enabled computation graphs (e.g., pytorch, tensorflow, jax) and standard scientific packages (e.g., cv2, scikit-image, etc.)
Hardware Requirements	CUDA-enabled PC/Laptop/Server with at least 8GB VRAM (recommended 12GB)
Status of the development of the component	Partially developed (TRL 3)

Table 17: SUN Components Description - Semantic 3D Scene Reconstruction

WP and Task reference	T3.5 - AI to learn objects for the virtual world
Responsible	Fabio Carrara
Name of Component/Service:	Semantic 3D Scene Reconstruction
Type	*Software*
Functionality	The component reconstructs a rough 3D scene enriched with querable semantic embeddings extracted from foundation vision-language AI models. Given an RGBD video and corresponding camera poses (usually provided by an XR device), this component constructs a 3D model that can be queried with textual (e.g., a brief description) or visual (e.g., an image crop) queries. As a result, the points and parts of the 3D model matching the query are highlighted, providing a rough localization and segmentation of the desired scene.
Input Connections & Interfaces	- posed RGBD frames from sensor stream, probably passed via API (theoretically input from sensors would be possible, but HW requirements hardly will fit in the possibilities of XR device). - query data (e.g., text, images, other) via API

Output Connections & Interfaces	Outputs (e.g., query results, pixel-aligned features, other TBD depending on interactions) available via API Possible consumers: 3D reconstruction pipelines? They could fuse extracted features into queriable 3D assets.
Software Requirements/Development Language	OS or containers (Docker, podman) for running Python scripts with CUDA-enabled computation graphs (e.g., pytorch, tensorflow, jax) and standard scientific packages (e.g., cv2, scikit-image, etc.)
Hardware Requirements	Server with at least a CUDA-enabled GPU with 10GB+ VRAM
Status of the development of the component	Partially developed (TRL 3)

Table 18: SUN Components Description - EMG decoding system for hand and wrist kinematics

WP and Task reference	WP3
Responsible	EPFL – Vincent Mendez (Vincent.mendez@epfl.ch)
Name of Component/Service:	EMG decoding system for hand and wrist kinematics
Type	Hardware and Software
Functionality	A custom-made EMG (Electromyography) system (or commercial one if needed) and software to decode finger and wrist kinematics based on muscular activity. Force levels could also be added.
Input Connections & Interfaces	Recording of a participant.
Output Connections & Interfaces	TCP/IP socket sending one value between 0 and N where N is the number of hand poses decoded required by the SUN platform.

Software Requirements/Development Language	Software implemented in Python. No dependencies are required as values are sent on the network on an adaptable communication protocol.
Hardware Requirements	The device consists of Ag/AgCl electrodes connected to an amplifier, two IMUs connected to a microcontroller, and a webcam for calibration. Only a computer with sufficient computing power is necessary (the minimal computing power was not evaluated but a laptop with a simple GPU is enough).
Status of the development of the component	Hardware is fully integrated, and software requires data to train the machine learning model. Optimization of decoding performance has to be performed.

Table 19: SUN Components Description - SUN Components Description

WP and Task reference	WP4 - Collaborative 3D acquisition and real-time XR visualization T4.1 - XR multimodal collaborative solutions
Responsible	TUC Katerina Mania amania@tuc.gr
Name of Component/Service:	AR App Pilot 1 (Collaboration, Glanceable Interfaces)
Type	*Software*
Functionality	AR application for pilot 1 upper limb use case, AR application for pilot 1 lower limb use case. It will include the necessary interfaces (UIs) and provide the functionalities to perform the rehabilitation tasks in the AR environment.

	• Interaction system: hand and gaze input configuration for UI and object interaction • Avatar Integration: it serves as a guiding entity, displaying and/or replicating user movements • Augmented feedback via visual cues based on data from T3.2 • Collaboration mechanisms with user connection and action/ object synchronization • Gamification elements
Input Connections & Interfaces	• Processed data from the postural assessment component -T3.2. via pub/sub protocol • 3D poses from the pose estimation component - T3.3 via pub/sub protocol • Avatar - T3.4 (preloaded input)
Output Connections & Interfaces	Generated events based on virtual object interaction or directional hints to Haptic Components - T3.1 via API. The data will be provided via Wifi network.
Software Requirements/Development Language	Development platform: Unity, Programming Language: C#, MRTK SDK for AR/VR development
Hardware Requirements	Microsoft's Hololens 2 HMD device
Status of the development of the component	TRL 2 - technology concept formulated

Table 20: SUN Components Description - Mass distribution acquisition device

WP and Task reference	WP4 – Task 4.2 Acquisition of physical properties for 3D objects
Responsible	CNR – Gianpaolo Palma

Name of Component/Service:	Mass distribution acquisition device
Type	Hardware and software
Functionality	The hardware component is in charge of the automatic acquisition of the data need for the estimation of the mass distribution. It is composed by a passive gripper sensorized with pressure sensors on the fingertips and a set of cameras that acquire the images during the manipulation of the object by the gripper. The main functionalities are: • the manipulation of the object by rotation of the gripper by means of a stepper motor; • the acquisition of the pressure sensor data and the camera image for each rotation step. The software component is composed by: • a tool to manage the hardware device (to start the automatic acquisition and to get the data) • the module to elaborate on the acquired data (images and sensor data) and estimate the mass distribution.
Input Connections & Interfaces	The component gets the input data from the hardware devices (pressure sensors, stepper motor, and cameras). The hardware is managed by two microcontrollers synchronized by a software tool via USB with an ad-hoc communication protocol.
Output Connections & Interfaces	The data acquired by the hardware device are elaborated by a software module that estimates the mass distribution. The estimated mass distribution is saved on a text file ready to be used from other components.

Software Requirements/Development Language	Programming languages: - C++ (software) - Arduino C++ variant (microcontroller) Required libraries: - Qt - OpenCV
Hardware Requirements	The module requires the ad-hoc hardware developed in the task. In the specific: - the passive gripper sensorized with the pressure sensors; - the stepper motor to manipulate the gripper; - 4 action cams for the image-based acquisition (GoPro Hero 5 or next); - a plywood box to mount the cameras and to allow their marker-based calibration; - 2 microcontrollers to manage the stepper motor, the pressure sensor, and the camera
Status of the development of the component	Partially developed

Table 21: SUN Components Description - Hololight Stream

WP and Task reference	WP4, T4.3
Responsible	HOLO
Name of Component/Service:	Hololight Stream

Type	*SDK (Software Development Kit)*
Functionality	Hololight Stream utilizes the protocol WebRTC for remote application rendering of any XR Unity application. Once used the application becomes a two-component solution. - The server Unity application, that contains all app elements, the logic, etc. running on a Windows machine or the Cloud. - The client application, installed on the smart glasses transmits data like SLAM, head pose, microphone input, etc.
Input Connections & Interfaces	WebRTC Connection via port 9999 Specific input depends on the developed application.
Output Connections & Interfaces	Outgoing UDP Ports 16384-32768 Specific output depends on the developed application.
Software Requirements/Development Language	Visual Studio Components: Universal Windows Platform, Game development with Unity, Game development with C++, Desktop Development with C++ Unity Version: 2021.3.x (minimum) Unity Components: Universal Platform Build Support, Windows Build Support IL2CPP Mixed Reality Toolkit Version: 2.7.x, 2.8.x Languages: C# or C++ when used with DirectX
Hardware Requirements	**Recommended Values** Operating System: Windows 10 (10.0..17763 Build), Windows 11, Windows Server 2019 Memory: 64 GB CPU: Intel i7 12 Gen. 12 Cores, AMD Ryzen 9 3900X GPU: NVIDIA RTX 3080 TI, NVIDIA GRID for VMs

	Storage: SSD or NVMe Network: Wi-Fi 5Ghz Bandwidth: 40 Mbit Round Trip Time (Latency): max. 50 ms
Status of the development of the component	Available SDK with TRL 7, SUN specific features in development

Table 22: SUN Components Description - NERF 3D reconstruction

WP and Task reference	WP4, T4.4
Responsible	CERTH (Panagiotis Vrachnos, Spyridon Symeonidis)
Name of Component/Service:	NERF 3D reconstruction
Type	Software
Functionality	This component makes use of Neural Radiance Fields (NeRFs) to generate 3D representations (of objects, environments, or avatars) from 2D images. NeRFs are based on deep neural networks in which a continuous volumetric scene function is learned to assign a colour and volume density to any point in the space. The weights of the network are optimized to encode the scene representation, enabling the model to easily render novel views (not explicitly available in the input data) observed from any point in space. A preliminary list of functions is the following: 1. Data acquisition thoroughly covering the scene of interest 2. Structure from Motion (SfM) techniques for camera pose estimation 3. Model training for a specific scene 4. Creation of custom camera paths

Input Connections & Interfaces	Input is a set of multimedia (images, videos) related to the element to be represented. The multimedia sources are to be defined; the initial experiments will be based on freely available online data.
Output Connections & Interfaces	The output format will be: • 2D renders (e.g., MP4, PNG, JPG file) • 3D textured geometry (e.g., ply file) • 3D point cloud (e.g., ply file) • Custom camera poses (e.g., json file) An API will be developed to provide the results to the SUN platform.
Software Requirements/Development Language	**Programming Language:** Python >= 3.7 **Tools/Libraries:** PyTorch, OpenCV, FFMPEG
Hardware Requirements	**Minimum system requirements:** Operating system: Windows 10 / Ubuntu 20.04 RAM: >=8GB Free disk space >=100GB GPU-RAM >= 12GB (e.g., NVIDIA GeForce RTX 3060)
Status of the development of the component	The component will be developed from scratch for the SUN project. Starting TRL: 2

Table 23: SUN Components Description - AI COMPLETION OF 3D MODELS

WP and Task reference	4.4
Responsible	Marco Callieri – task leader
Name of Component/Service:	AI COMPLETION OF 3D MODELS
Type	software
Functionality	The main aim is to provide ways to complete, enhance geometry, and texture of 3D models, using AI methods.

	There will be multiple components, each one targeting a specific kind of processing, filtering, data generation. Some of the components will work inside the 3D creation pipeline (photogrammetry), using AI to help the production of better 3D models. Some other components will work on existing 3D models and apply specific, AI-based filtering and processing to obtain better geometry and texture.
Input Connections & Interfaces	The components are basically data source: they help in the production of assets. The input of these components is either raw image data to be processed using photogrammetry, or existing 3D models to be enhanced.
Output Connections & Interfaces	The components do not directly interact with other project components, and act as stand-alone tools. The 3D data produced by the components will be available to be used by the project sw/hw as 3D assets (3D model files with texture).
Software Requirements/Development Language	As some of the components will work inside the photogrammetry pipeline, they will only work in conjunction with photogrammetry software. We will mostly work with C++; Python ad other scripting languages will be used to automatize tasks.
Hardware Requirements	The components do not require any specific hardware; however, as most of the computation in this class of algorithms (photogrammetry, deep learning) happens on GPU, the components will need to run on PCs with an up-to-date GPU.
Status of the development of the component	In development.

Table 24: SUN Components Description - Fused Conditional Denoising Diffusion Probabilistic Model (FCDDPM)

WP and Task reference	WP 4 - Task 4.5
Responsible	CNR - Marco Di Benedetto
Name of Component/Service:	Fused Conditional Denoising Diffusion Probabilistic Model (FCDDPM)
Type	Software
Functionality	The component will be able to reconstruct, as a 3D mesh, a relatively large environment, overcoming the limits of short-range depth cameras or sparse long-range scanners. The component will use techniques from Generative AI models and an architecture setup able to interpret and produce signed distance fields (SDF) volumes to generate the final mesh.
Input Connections & Interfaces	The component API will take as input N couples of temporally ordered RGB and its semantic-segmented image acquired from a large environment, and the corresponding reconstructed sparse point cloud (e.g., 3D point features from a SLAM process).
Output Connections & Interfaces	The component will output a 3D mesh of the acquired environment. The mesh is extracted from the neural network-generated SDF volume.
Software Requirements/Development Language	Programming Languages: – Python v3+

	Required Libraries: – PyTorch v2+ – NumPy – Scikit Learn – PyMeshLab
Hardware Requirements	A CUDA-enabled processing unit (e.g., NVIDIA 4060 RTX for desktops, or NVIDIA Jetson for embedding devices).
Status of the development of the component	Partially developed (starting TRL = 3 / 4).

Table 25: SUN Components Description - Tokenized Platform

WP and Task reference	T5.3
Responsible	ENG – Ferdinando Bosco
Name of Component/Service:	Tokenized Platform
Type	Software
Functionality	The Tokenized Platform, based on blockchain technology, will exploit fungible and non-fungible tokens (NFTs) for digital assets transactions and unique identifications. The Tokenized platform will allow the platform the creation and trading of digital tokens implementing the entire digital assets value chain: creation, distribution, and revenue models. The blockchain technology will ensure transparency and security, registering all the assets transactions in a decentralized way.

Input Connections & Interfaces	The Tokenized Planform aims to receive information/data from any tools, components, or devices that allow the platform the creation of digital assets that need to be managed and exchanged among different stakeholders. The Tokenized Platform will expose standard Open APIs for the interconnections with blockchain infrastructure and smart contracts. In addition, the Tokenized Platform can manage other protocols for different input connections if needed (e.g., MQTT)
Output Connections & Interfaces	The output of the Tokenized Platform will be available through UI (for a complete digital asset management) and APIs (for a possible integration with other tools).
Software Requirements/Development Language	The Tokenized Platform will be managed in a cloud environment and offered as-a-service. In alternative, the Tokenized Platform can be released as a Docker App and configured for using external blockchain infrastructure (e.g., Ethereum and/or Polygon networks)
Hardware Requirements	No particular hardware requirements have been identified so far.
Status of the development of the component	The Tokenized Platform will be implemented using ChainPro, a Blockchain-based Framework developed in ENG R&I. ChainPro offers basic features for quick and easy delivery of blockchain-based applications. The TRL starting level can be considered as 4-5.

Table 26: SUN Components Description - Cyber threat detection

WP and Task reference	5.3
Responsible	UoG, George Loukas

Name of Component/Service:	Cyber threat detection
Type	Software
Functionality	Monitors the operation of an XR system to spot signs of a cyber security breach and warn the user accordingly before its impact escalates.
Input Connections & Interfaces	Sensors on the head-mounted display (accelerometers, gyroscopes, magnetometers, microphones, eyetracking sensors, GPS, optical sensors, proximity sensors), network traffic metrics, GPU metrics The input is currently received from the VR head-mounted display via a custom script created in the Unity game engine. This script is currently designed to extract data features in real-time while a user is immersed in VR and stored in a file. In the case of AR, the input is expected to be received from the built-in sensors and cameras in the headset, but we will know only after we choose an AR device as a consortium. For the Machine learning deployment platform, we are evaluating whether to deploy on AWS, our own server. We also have the option of using an Android 10 OS like the Quest 2 headset because it is possible to deploy TensorFlow Lite models directly on the headset itself. We have not tested this yet. We are still doing the machine learning offline.
Output Connections & Interfaces	Primarily the display device itself, but we have the option of sound and haptics too. We will most likely develop a

	basic UI for visually warning the user, but this is subject to research as we aim to not unduly disrupt the immersion and presence of the XR experience. In this early stage, the data monitored from the different sensors, GPU, and network is stored in a simple .csv file, which is then fed offline into a machine learning module. The resulting output which is again a file consisting is sent back to the Unity game engine to warn the user.
Software Requirements/Development Language	C# and Python
Hardware Requirements	Assuming a single device for both VR and AR/MR, then there are not many options. The commercial leaders are: Vive XR Elite and Meta Quest Pro. The Quest Pro looks more attractive because it runs Android.
Status of the development of the component	Partially developed for VR. TRL 4. The monitoring will be from scratch for AR/MR, but we will re-use the VR machine learning with suitably modified models.

Table 27: SUN Components Description - Transcranial temporal interference stimulation (tTIS) for the non-invasive stimulation of deep brain structures.

WP and Task reference	WP6 – T6.2
Responsible	EPFL – Pierre Vassiliadis
Name of Component/Service:	Transcranial temporal interference stimulation (tTIS) for the non-invasive stimulation of deep brain structures.
Type	Hardware and software

Functionality	This component is responsible for the non-invasive brain stimulation of the patient at specific timings. tTIS currents are generated by bipolar constant current stimulators (Digitimer). The stimulation patterns are created using a custom-written MATLAB-based (MathWorks) graphical user interface (GUI) and transmitted to the current sources. Current amplitude and frequency can be specified through the GUI.
Input Connections & Interfaces	This component receives input from the SUN platform using a TCP socket to inform on specific events happening in the VR app. Brain stimulation is triggered on specific events. The events are received by a custom MATLAB script and stimulation patterns created in the MATLAB environment are transmitted to the stimulators using a standard digital-to-analog converter (DAQ USB-6216, National Instruments).
Output Connections & Interfaces	The output of the stimulators is connected to two pairs of conductive rubber surface electrodes applied on the head. An audio transformer is added between stimulators and subjects, in order to avoid possible direct current accumulation.
Software Requirements/Development Language	Custom-written MATLAB-based (MathWorks) graphical user interface to create the stimulation patterns.

Hardware Requirements	The currents are delivered by two independent DS5 isolated bipolar constant current stimulators (Digitimer). A computer with sufficient computing power (not evaluated) is necessary to set the stimulation parameters through the GUI. Audio-transformer between stimulators and subjects is needed to prevent current accumulation.
Status of the development of the component	The component is fully developed, and it has already been tested and used. Ethical approval is in place to test it in the context of VR.

Table 28: SUN Components Description - AR App – Pilot 2

WP and Task reference	WP 6, T6.2
Responsible	HOLO
Name of Component/Service:	AR App – Pilot 2
Type	*Software*
Functionality	The application will be split into 2 sections. The 1st section will cover the training and practice part of the pilot. Specific training steps for the correct application of PPE will be visualized through the AR application. The 2nd section will cover the task assignment and situational awareness aspect of the pilot. The application will interface with the object recognition tool, a task management algorithm, additional gesture recognition tool, and transmit the camera stream from the HoloLens 2 to external solutions.

	To enable the data heavy visualization of the training, as well as the camera transmission of the HoloLens 2, the Hololight Stream SDK will be integrated.
Input Connections & Interfaces	REST APIs, pub/sub, TCP/IP, and other interfaces compatible with Unity
Output Connections & Interfaces	WebRTC for video transmission
Software Requirements/Development Language	Any interface must be compatible with Unity.
Hardware Requirements	**Recommended Values** Operating System: Windows 10 (10.0..17763 Build), Windows 11, Windows Server 2019 Memory: 64 GB CPU: Intel i7 12 Gen. 12 Cores, AMD Ryzen 9 3900X GPU: NVIDIA RTX 3080 TI, NVIDIA GRID for VMs Storage: SSD or NVMe Network: Wi-Fi 5Ghz Bandwidth: 40 Mbit Round Trip Time (Latency): max. 50 ms
Status of the development of the component	Architecture created, first prototype in development

Table 29: SUN Components Description - Room Scan App

WP and Task reference	WP6, T6.2
Responsible	HOLO
Name of Component/Service:	Room Scan App
Type	*Software*

Functionality	The room scanning app allows users to scan their environment and add spawn points, such as locations where objects/avatars would appear. After scanning the room, users must scan a QR code to establish an origin point for another application. This other application can later scan the same QR code, which should remain in the same position, to re-reference the room and its spawn points. This process involves loading a saved .gltf file that contains the room's mesh and the defined spawn points. When the QR code is detected by another device, the .gltf file is imported and aligned based on the QR code's position and rotation. Essentially, the XR Glass detects the QR code's position and rotation, allowing developers to position the room and spawn points relative to the QR code's pose.
Input Connections & Interfaces	SLAM data from the HoloLens 2 sensors.
Output Connections & Interfaces	Mesh visualizing the generated scan stored as .gltf file.
Software Requirements/Development Language	-
Hardware Requirements	HoloLens 2
Status of the development of the component	Completed Application – not a main tool of the project, but necessary to develop the Pilot 2 solution

Table 30: SUN Components Description - XR environment for pilot 3

WP and Task reference	WP6, T6.2
Responsible	ENG - Luca Greci luca.greci@eng.it

Name of Component/Service:	XR environment for pilot 3
Type	*Software*
Functionality	The environment is a multi-user virtual space where people can connect, interact, move, and communicate. The technologies used include the Netcode framework and Vivox, services provided by Unity Game Services. Netcode manages the synchronization of actions among users, while Vivox provides real-time bidirectional voice communication features, allowing players to communicate. In the XRE, users can communicate and interact via avatars, as well as interact with virtual objects within the space." Functions list: Multi-user connection Action synchronization Real-time voice communication User Avatar interaction Interaction with virtual objects
Input Connections & Interfaces	Input from BMMIs (string) and tokenized platform (.glb) via API
Output Connections & Interfaces	Output to Temperature feedback device via API: string
Software Requirements/Development Language	Programming Language: C# Platform: Unity 3d

	Tools/Libraries: Netcode, Vivox, Text to speech service (Google or AWS), Meta SDK
Hardware Requirements	Meta HMD device Minimum system requirements: Operating system: Windows 10 RAM: >=8GB Free disk space >=100GB GPU-RAM >= 12GB (e.g., NVIDIA GeForce RTX 3060)
Status of the development of the component	The component will be developed from scratch for the SUN project. Starting TRL: 2

Table 31: SUN Components Description - Logistics Processes Optimization

WP and Task reference	WP6 T6.2
Responsible	UPV, Josefa Mula <fmula@upv.es>
Name of Component/Service:	**Logistics Processes Optimization**
Type	Software
Functionality	The component provides task prioritization and resource allocation to ensure seamless operations on the manufacturing floor. Using information from real-time monitoring and internal systems, it creates a prioritized list of tasks necessary to keep production efficiency high.
Input Connections & Interfaces	The component relies on input from the end user Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES). These provide information about production orders, inventory levels, and machinery status. Query the system information via API.

Output Connections & Interfaces	The component sends the tasks priority information to the XR headset UI system for display. This provides real-time information on critical tasks to be completed, helping workers make decisions and execute their responsibilities within the shopfloor.
Software Requirements/Development Language	**Software requirements** System prepared for container integration (Docker, etc.), or running Python scripts. **Development Language** Python
Hardware Requirements	Computation server
Status of the development of the component	To be developed from scratch. TRL 2

4.4 Collaborative Workshops

As described in section 4.1, a key approach of the methodology used for the SUN architecture design was to match the top-down approach in which a detailed analysis of technical components was conducted and an initial reference architecture was proposed with the bottom-up approach in which the use cases, requirements, and specifications from the SUN Pilots was collected. The intersection point between these two approaches was facilitated through collaborative architectural workshops involving end-user partners, pilot drivers, and technical partners in charge of the technical component research and development. The workshops allowed us to refine the SUN Architecture iteratively and collaboratively. The workshops were prepared and facilitated by ENG and IN2.

The workshops followed a common structure and a number of steps:

- A preparation phase before the workshops during which the objectives of the mapping exercise were clearly defined. During this phase, ENG and IN2 as organisers collected and prepared all relevant documentation, such as lists of the technical components and the reference architecture. This information served as the foundational material for discussions and was distributed to all participants to ensure a uniform understanding of the current SUN platform and the architectural goals.

- ENG and IN2 then carefully designed the workshop, preparing the structure, agenda, and tools to be used so that there was active participation from both technical experts and user representatives.
- During the workshop different facilitation techniques were used to encourage participants to think critically about the technical and operational aspects of the platform and how these could be best integrated into the reference architecture.
- Feedback and insights were collected systematically through various means, including notes, and audiovisual recordings. This data was then analyzed to identify key themes, challenges, and opportunities for better alignment with the reference architecture.
- Sharing progress and iterating. Findings from each workshop were documented and shared with all partners and formed the basis for the next iteration/workshop. This iterative approach ensured that the final mapping of technical components to the reference architecture was both comprehensive and consensual among all parties involved.

4.4.1 First Collaborative Workshop – Components identifications and mapping

The first collaborative workshop took place during the Plenary Meeting at Malta and involved both user partners and technical partners.

The main input for this workshop was the preliminary list of tools and components collected from technical partners. In the first iteration, a list of 18 items was identified and for each of them a technical responsible and technical specifications were defined.

During the workshop, the different tools and components were described to the whole consortium and a mapping between items and pilots was done thanks to the support of the user partners.

Additionally, some preliminary technical details and specifications were identified to pave the way for the first draft of the SUN Architecture. In particular, for each item, were identified the:

- Technical Description
- Technical Responsible
- Pilot application
- Preliminary specifications, dependencies, and interactions

4.4.2 Second Collaborative Workshop – First Draft of Architectural and Pilot Mapping

Following an iterative approach, the outcomes of the first workshop conducted to a first draft of a conceptual architecture, in which the first core elements were identified.

The first draft of the SUN Architecture highlighted the need to have a modular architecture with several layers able to interact with each other to ensure the

implementation of the necessary services as well as the possibility to include data from devices and sensors.

For this reason, two main layer categories were initially defined:

- Architectural Layers, including all the components and tools, divided by characteristics and functionalities.
- Communication Layers, intermediate layers able to connect the architectural layers and the components on top of them.

In addition, it was also identified the necessity to have two vertical components, shared in the overall platform for ensuring data storage, access management, and cybersecurity.

Finally, particular attention was paid to the "Creation Tools", any kind of tool able to create XR content to be shared in the overall SUN Platform.

The second collaborative workshop was held during the Plenary Meeting in Valencia and was divided into three main activities:

1. Presentation of the first draft of the SUN Architecture
2. Mapping of Components within the SUN Architecture
3. Toward integration - Mapping of Components in different use cases and scenarios

Presentation of the first draft of the SUN Architecture

During the first part of the workshop the draft of the SUN Architecture was presented, the technical layers described, and some open points discussed. In particular:

- Which Data must/should be stored?
- Which protocols must/should be used at each level?
- Is there any SUN component, tool, or service not mappable in the architecture?

Mapping of Components within the SUN Architecture

During the second part of the workshop, the list of components was revised (starting from the outcomes of the first workshop) and all the components were mapped within the SUN Architecture. To make this, each technical partner, using a common Jam Board was able to identify the correct layer for its own components.

The result of this activity, shown in Figure 14, demonstrated that the first draft of SUN Architecture was already representing a modular architecture able to include all the possible SUN components.

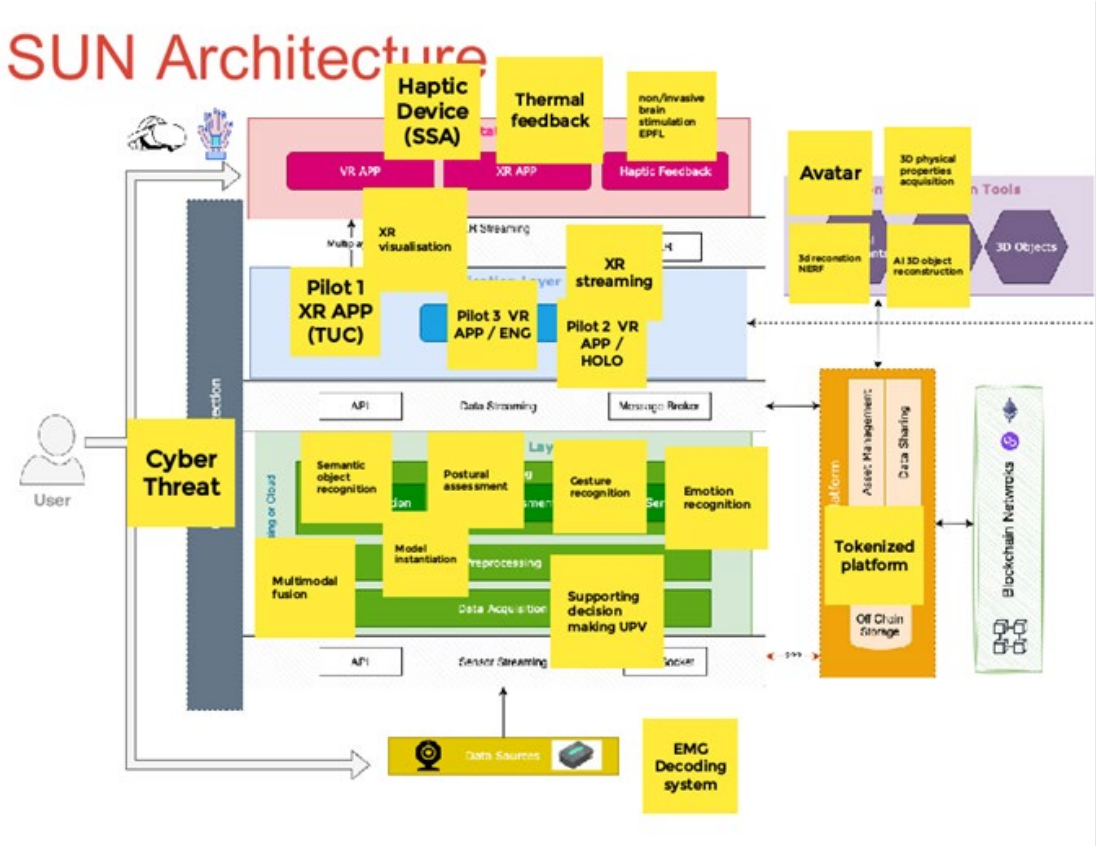

Figure 14: Mapping of SUN Components with Architecture - First Iteration

Toward integration - Mapping of Components in different use cases and scenarios

The last phase of the workshop focused on the integration phase. In fact, continuing the work done in the first workshop, the components were mapped in the different pilot implementations both at the scenario and use case level, based on the use cases defined and requirements collected until that moment.

To make this, many Jam Boards were created and for each of them, the technical partners and the pilot responsible selected all the components to be used and integrated. Figure 15 shows the board for Pilot 1 and Use Case 1 as an example.

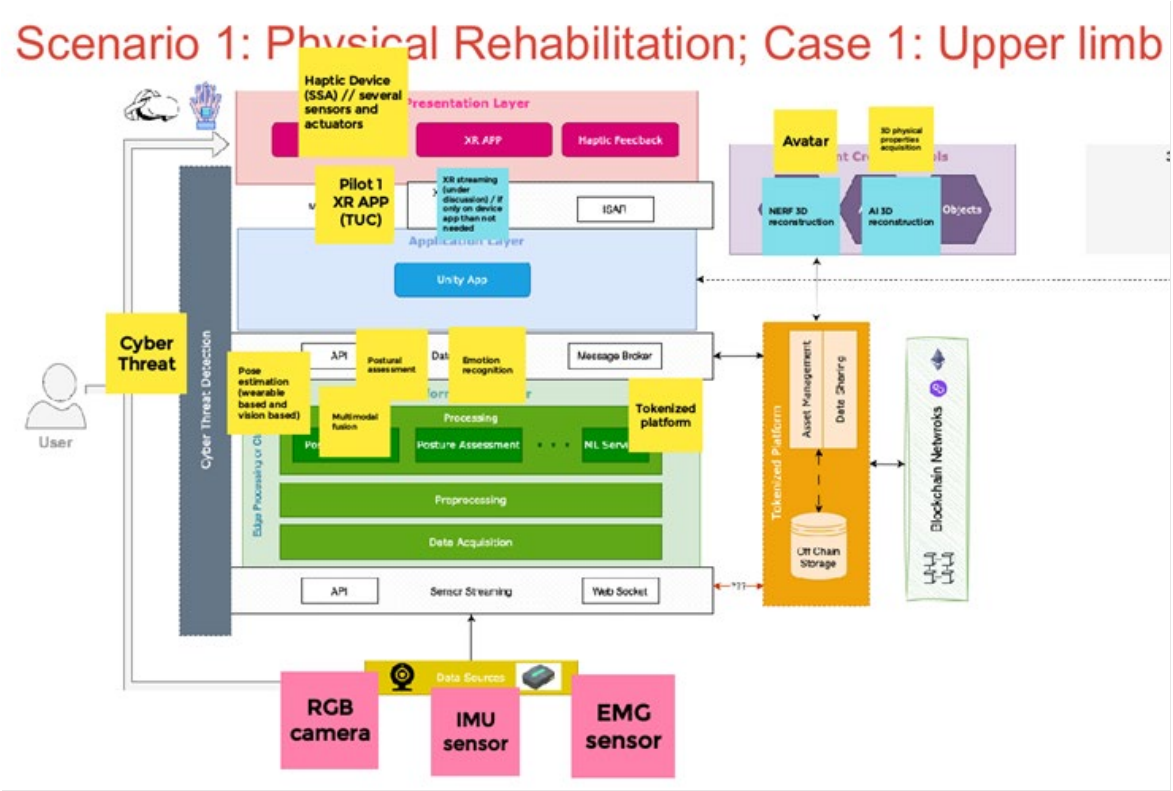

Figure 15: SUN Components Pilot Mapping - Example

In particular, six specific boards were created:

- Scenario 1: Physical Rehabilitation; Case 1: Upper limb
- Scenario 1: Physical Rehabilitation; Case 2: Lower limb
- Scenario 2: Industry 5.0; Case 1: PPE Detection
- Scenario 2: Industry 5.0; Case 2: Safety and Object Tracking
- Scenario 3: Cerebral Rehabilitation; Case 1: Stroke / Tetraplegic
- Scenario 3: Cerebral Rehabilitation; Case 2: Apathy

4.4.3 Final Workshop – Consolidation and Components Interactions

The final round of collaborative workshops was held online and in three separate sessions, one for each pilot.

The final workshops focused on three main objectives:

1. Consolidation of components lists and mapping within the different pilots (use cases and scenarios).
2. Definition of interaction among components within each pilot (connections, data exchanged, etc.)
3. Identification of communication, integration, and deployment requirements

Consolidation of components lists and mapping within the different pilots (use cases and scenarios).

This first activity allowed us to consolidate the overall list of components, already reported in Table 4, as well as the subset of them to be used within each pilot. The mapping of the components will be reported in the specific paragraphs of Section 5.3.

Definition of interaction among components within each pilot (connections, data exchanged, etc.)

The second step involved again the technical partners and allowed them to identify within each pilot architecture, the interactions among components, to be used as input for the identification of architectural and technical specifications of the SUN Platform.

Figure 16 reports, as an example, the outcome of Pilot 3, depicted using Draw.io tool.

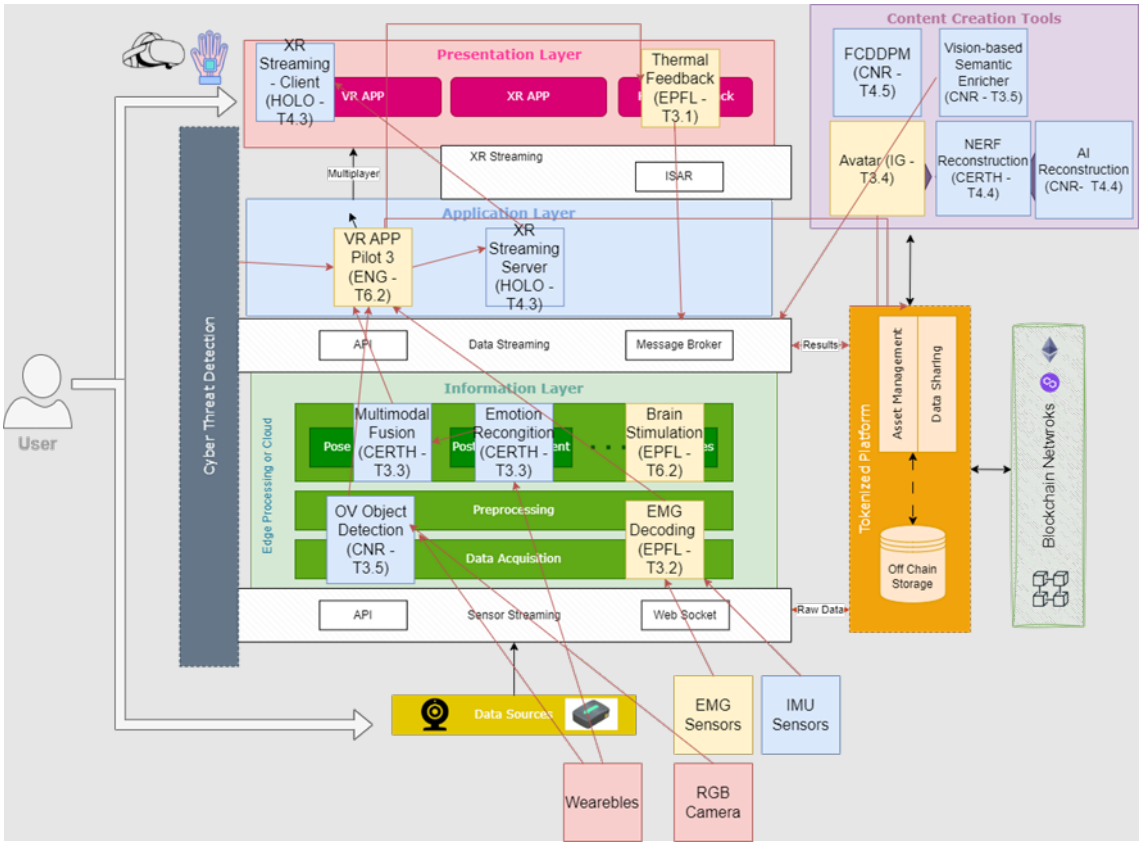

Figure 16: SUN Components Mapping and Interactions identification – Pilot 3 Example

Identification of communication, integration, and deployment requirements

The final step of this process was held offline. Starting from the components mapping in step 2, a list of interactions was elicited and shared with technical partners that supported the identification of additional technical details about component interactions.

For each interaction, the following information was collected:

- **Communication mechanisms**, technology, and protocols for implementing the interactions.
- **Data Exchanged**, high-level data to be exchanged during the components' interactions.
- **Type of interaction**, additional information about components interactions, such as frequency, type of request (live or on-demand), etc.

A complete list of interactions, identified for each pilot, is reported in Section 5.3.

5 SUN Reference Architecture

5.1 Introduction

The purpose of the Reference Architecture is to provide guidance for design and technical developments, incorporating the vision of the solution, as well as the requirements and the technical specifications. It can be intended as the shared baseline for implementing all the possible systems that will be part of the SUN XR Platform.

As described in 4, starting from the results collected in WP2 about user requirements and scenarios, together with the components specification elicited in WP3 and WP4, a high-level system architecture and functional specification of the SUN XR Platform was defined.

Furthermore, following the insights of the state-of-the-art analysis, particular attention was paid to four key aspects:

- Interoperability and Standardization
- Scalability for Large Multi-User Environments
- Real-Time Data Integration and Collaboration:
- Security and Privacy

5.2 Reference Architecture

The SUN Reference Architecture shown in Figure 17: SUN Reference Architecture - Final Version, represents the main outcome of the iterative and collaborative activities described in Section 4.

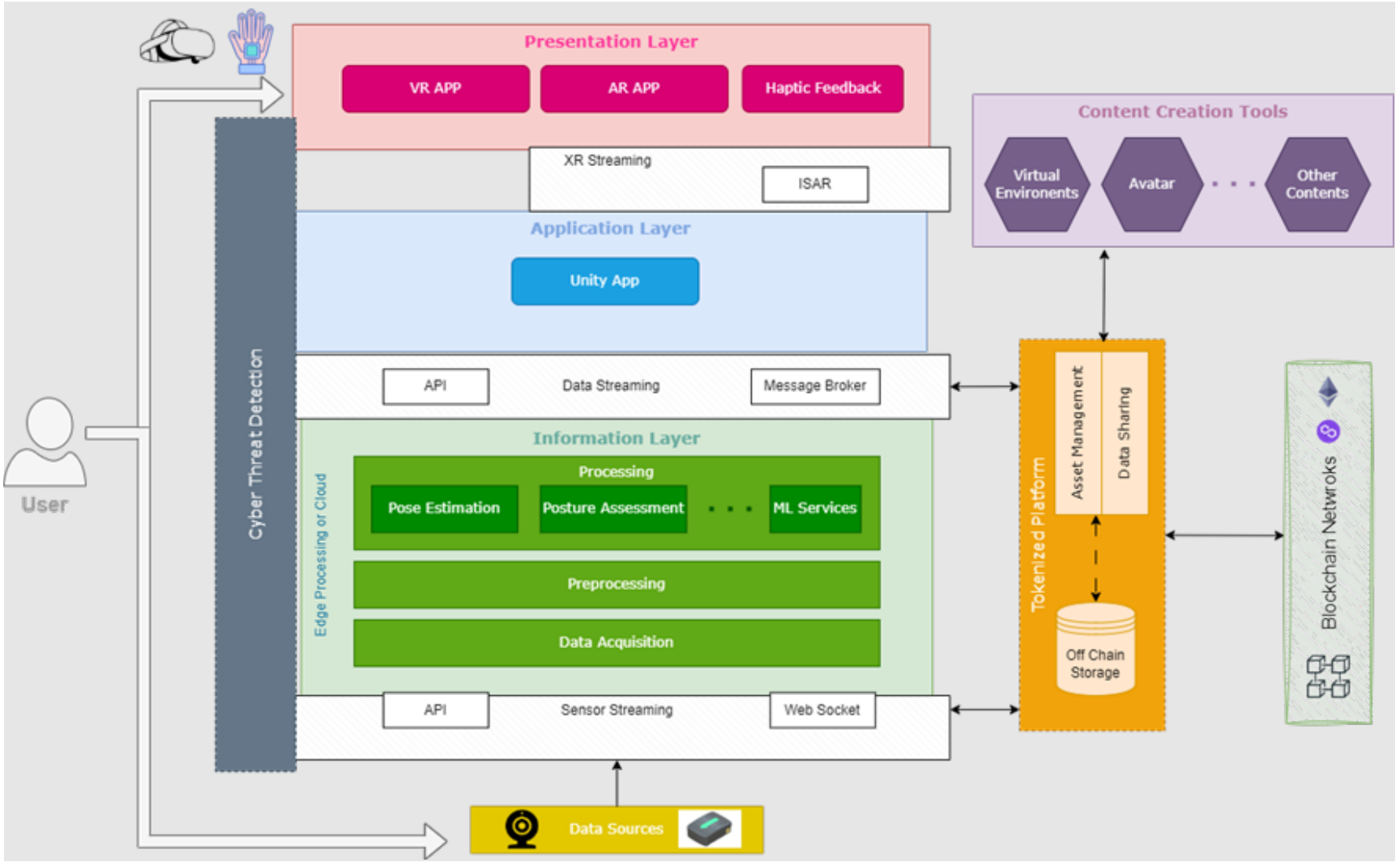

Figure 17: SUN Reference Architecture - Final Version

The SUN Reference Architecture is depicted as a multi-layer architecture, allowing the development of modular solutions and leveraging on interoperability aspects.

In particular, two types of layers were identified:

- **Architectural Layers**, including all the components and tools, divided by characteristics and functionalities.
- **Communication Layers**, intermediate layers able to connect the architectural layers and the components on top of them.

The Architectural Layers (Information, Application, and Presentation) are the core part of the SUN Reference Architecture, in which all the XR components can be developed and integrated, respecting their own main characteristics.

Based on the specifications of each XR component, the different layers provide different characteristics, to ensure maximum flexibility and scalability of the solution.

It is important to emphasize that the Application and Presentation Layer, can be separated thanks to the innovative solution of the XR Cloud streaming, which allows us to run any kind of XR Application on a standard device (e.g., a laptop) and use the XR device only for the presentation to the end-user, optimizing its performance.

The Communication Layers (Sensor, Data, and XR streaming) have different roles and capabilities, depending on the needs of the connected Architectural Layers. They implement and integrate different protocols, ensuring at any level of the process the best performance.

In addition, it was also clear the necessity to ensure security and privacy aspects during the data sharing, and for this reason, two core components are shared in the overall platform for ensuring data storage, access management, and cybersecurity: the **Cyberthreat Detection and the Tokenized Platform**.

Finally, all the **Content Creation Tools**, were grouped as external tools since they can work offline and outside of the deployed platform. They can be easily connected through the Tokenized Platform, ensuring data access management and data sharing in a scalable and secure way.

In the paragraphs below all the building blocks are described more in detail.

5.2.1 Architectural Layers

5.2.1.1 Information Layer

The Information Layer is the main entry point for data collected from the data sources (Cameras, Wearables, other devices).

It consists of three levels: **Data acquisition, pre-processing, and processing**.

In these three levels, the data is acquired from the data sources, prepared, and processed for elaboration.

Typical data collected and elaborated in this layer is sensor and/or video streaming and the main outputs are used within the XR Applications.

The components that are part of this layer are usually deployed at local or edge level to improve the performance and the integration of "live stream" data.

5.2.1.2 Application Layer

The Application Layer is quite important in the SUN Reference Architecture since it differentiates the **XR Applications Business Logic** from the frontend application.

It is strictly linked to the Presentation Layer, in fact, a typical XR Application directly runs in an end-user device, performing both the core logic and the visualisation in the same device.

The SUN Architecture Application Layer allows us to consider the Business Logic of an XR Application (e.g., Unity App) as a separate entity, not only from a conceptual perspective but also in a practical way, since the SUN Architecture supports the possibility to "stream" data between **Application Layer (Server Application) and Presentation Layer (Client Application)** rendering any XR Unity Application computed in a remote environment, thanks to HOLO Streaming tool.

5.2.1.3 Presentation Layer

The Presentation Layer consists of the main entry point for the end-users. It includes all the **frontend XR Applications** as well as additional **devices** that provide feedback to the users (e.g., Haptic or Thermal feedback).

All the applications, tools, and devices that interact with the end-users, are implemented at this level.

5.2.2 Communication Layers

As described in the previous paragraph, the SUN Reference Architecture implements three different communication layers, each of them with different characteristics, capable of optimizing communication between two adjacent layers.

5.2.2.1 Sensor Streaming

The Sensor Streaming Layer supports the communication and integration of external devices (data sources) with components deployed within the Information Layer.

This layer must perform very well, due to the kind and amount of data exchanged. Typical protocols and components part of this layer are **Web Socket and Message Broker**.

In addition, the Tokenized Platform can also interact with Sensor Streaming to give persistency to the raw data for further analysis.

5.2.2.2 Data Streaming

Within the Data Streaming Layer, all the outputs provided from Information Layer components (such as estimated poses or emotions feeds) are provided to the XR Applications, to be elaborated and used for implementing functionalities for the end-user apps.

In addition, the Application Layer exploits the Data Streaming, for integrating any kind of XR Content stored within the Tokenized Platform.

Vice versa, the Data Streaming is also able to receive data from the XR Applications and store it into the Tokenized Platform or provide it to the Information Layer modules.

The Data Streaming Layer should support both synchronous and asynchronous data exchange and for this reason, it must implement both **REST API or Message Broker (MQTT or Kafka)**, depending on the kind of data and the needs of the application.

5.2.2.3 XR Streaming

The XR Streaming Layer is a peculiarity of the SUN Reference Architecture since it is not a typical communication layer in the XR Platform.

In fact, due to the innovative possibilities offered by HOLO XR Streaming, this layer can "stream" data between the Application Layer (Server Application) and Presentation Layer (Client Application) rendering any XR Unity Application computed in a remote environment. While traditional streaming technology focuses on remote rendering approaches to deliver 3D objects, HOLO's XR Streaming involves the streaming of the entire application. By rendering any AR or VR application as a whole, Hololight Stream is able to provide an environment that suits to every use case. With potentially unlimited performance, an agnostic approach for devices and server infrastructure, and a simple way of app integration, Hololight Stream offers various benefits for app developers as well as end users.

The **XR Streaming exploits the WebRTC** protocol that provides real-time communication (RTC) via simple application programming interfaces.

5.2.3 Data Storage and Cybersecurity

5.2.3.1 Cyber Threat Detection

The Cyber Threat Detection is a vertical component of the SUN Architecture with the role of monitoring the operation of an XR platform to spot signs of a cyber security breach and warn the user accordingly before its impact escalates.

It receives input from device sensors in real-time while a user is immersed in AR/VR environment. The resulting output can be sent back to the Unity game engine to warn the user in a real-time mode.

Additional details about Cyber Threat Detection and cybersecurity aspects are reported in Section 6.3.3.

5.2.3.2 Tokenized Platform

The Tokenized Platform, based on blockchain technology, plays a dual role in the SUN Architecture.

First, it exploits fungible and non-fungible tokens (NFTs) for digital asset management and data sharing.

The Tokenized platform will allow the system the creation and trading of digital tokens implementing the entire digital assets value chain: creation, distribution, and revenue models.

Blockchain technology will ensure transparency and security, registering all the assets transactions in a decentralized way.

Second, it acts as the storage layer of the entire SUN Architecture, allowing it to store off-chain (in a standard database) any kind of data collected on the Streaming or Data Layer, maintaining the data access management and data sharing capabilities in terms of security.

5.2.4 Content Creation Tools

Content Creation Tools play a fundamental role in enriching the SUN Architecture offering, ensuring innovative and state-of-the-art XR contents are used and integrated on the SUN Platform.

These tools are intended as any kind of components that can run externally from the platform, create new XR Contents, and make it available to the entire SUN Architecture via the Tokenized Platform.

The Tokenized Platform, in this case, acts as the digital asset management platform, ensuring the ownership and digital asset availability to the SUN Platform in a secure and trusted way, exploiting blockchain technology.

5.3 Pilot and components mapping

The mapping activities between components and pilots were conducted and finalized in the last Architectural Workshop, as reported in Section 4.4.3.

This activity allowed us to identify which components will be used on each pilot and use case and how each component interacts with others, ensuring the collection of all the technical specifications for SUN Architecture.

Table 32 reports the final mapping and then more details are reported for each use case in the following paragraphs.

Table 32: SUN Components Pilot Mapping

Component	Pilot 1.1	Pilot 1.2	Pilot 2.1	Pilot 2.2	Pilot 3.1	Pilot 3.2
AR App Pilot 1 (Collaboration, Glanceable Interfaces)	YES	YES				
EMG decoding system for hand and wrist kinematics					YES	YES
Wearable system for thermal feedback					YES	YES
High-performance Interactive Streaming	OPT	OPT	YES	YES	OPT	OPT
Haptic device: Distributed Wearable Haptic system for directional hints and cues	YES	OPT				
Haptic device: Wearable Haptic system for manipulation cues	YES					
Postural Assessment and monitoring of body kinematics	OPT	YES				
AI Reconstruction of 3D Models	OPT		OPT	OPT	OPT	OPT
3D reconstruction using NERF	OPT	OPT	OPT	OPT	OPT	OPT
IGOODI End2End Avatar Production Pipeline	YES	YES	YES		YES	YES
Fused Conditional Denoising Diffusion Probabilistic Model					OPT	OPT
Wearable Based Pose Estimation	YES	YES				
Camera Based Pose Estimation	YES	YES				
Gesture Recognition			YES	YES		
Emotion Recognition	YES	YES			OPT	OPT
Multimodal fusion module	YES	YES			OPT	OPT

Supporting decision making				YES		
Tokenized Platform	YES	YES	YES	YES	YES	YES
Cyber Threat Detection	YES	YES	YES	YES	YES	YES
Non-invasive brain stimulation						YES
Open-Vocabulary Object Detection			YES	YES	OPT	OPT
Semantic 3D Scene Reconstruction			OPT	OPT	OPT	OPT
Acquisition of physical properties for 3D objects	OPT					
VR APP for pilot 3					YES	YES
Holo AR App - Pilot 2			YES	YES		
Room Scan App			OPT	YES		

5.3.1 Pilot 1

As described in Section 4.2 Pilot 1 focuses on two scenarios: rehabilitation of upper limb, and rehabilitation of lower limb. Figure 18 shows the SUN Architecture mapping for Pilot 1.

Pilot 1 offers an A**R Application** for performing rehabilitation tasks in the AR environment. It will interface with **Haptic Devices** to provide haptic feedback to the users in real-time. The AR Application can run directly on the end-user device or in a standard device exploiting the **HOLO XR Streaming** application.

Many contents can be integrated within the VR APP including **3D Objects, 3D Objects Properties, and Avatars**.

Within the Information Layer, Pilot 1 plans to collect data from **EMG Sensors, External RGB Cameras, IMU Sensors, and Wearables**. These data will be processed by several modules: **Camera and Wearable Based Pose Estimation, Postural Assessment, Emotion Recognition, Heart Rate Estimation, and Multimodal Fusion**.

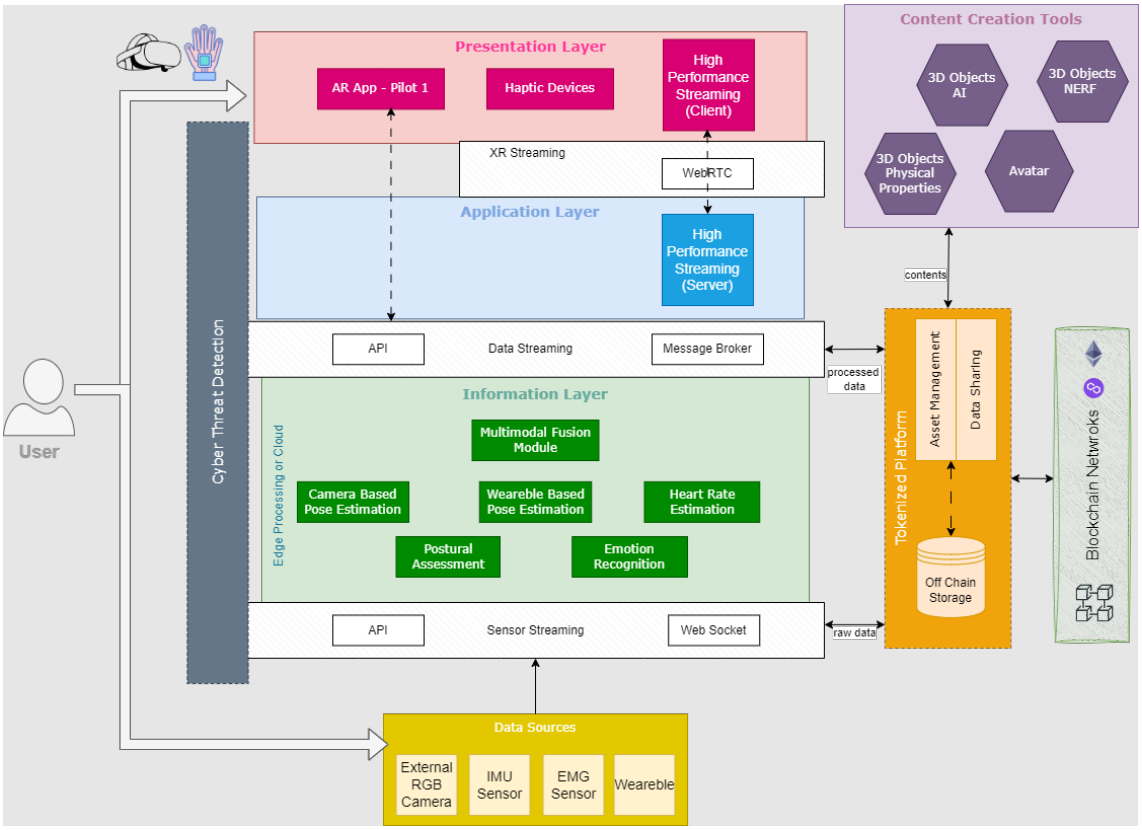

Figure 18: SUN Architecture Mapping for Pilot 1

Table 33 shows all the identified interaction among components within Pilot 1 Architecture.

Table 33: SUN Components Interactions - Pilot 1

Component A (Sender)	Component B (Receiver)	Communication Mechanism	Data Exchanged	Type of Interaction
Emotion Recognition	Multimodal Fusion Module	MQTT (Other solutions like REST, gRPC tbc)	Unimodal emotion recognition results	Messaging if MQTT is applied
Camera Based Pose Estimation	Multimodal Fusion Module	MQTT (Other solutions like REST, gRPC tbc)	Camera-based estimated poses	Messaging if MQTT is applied
Wearable Based Pose Estimation	Multimodal Fusion Module	MQTT (Other solutions like REST, gRPC tbc)	Wearable-based estimated poses	Messaging if MQTT is applied
Postural Assessment	AR APP Pilot 1	MQTT	Activity performed and Assessment (Correct/Wrong)	Pub/sub
Multimodal Fusion Module	AR APP Pilot 1	MQTT	Human poses and emotions	Pub/sub

AR APP Pilot 1	Haptic Device: directional hints	UDP (Preferably)	Game logic phase, pose (or high-level error est.)	XR:Pub Haptics/sub
AR APP Pilot 1	Haptic Device: manipulation cues	UDP (Preferably)	Contact events in the XR interaction	XR:Pub Haptics/sub
Avatar	Tokenized Platform	REST API	Avatars	Sync - on demand
NERF Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
AI Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
Acquisition of physical properties	Tokenized Platform	REST API	3d vector and 3x3 matrix	Sync - on demand
Tokenized Platform	AR APP Pilot 1	REST API	Contents	Sync - on demand
AR APP Pilot 1	Tokenized Platform	MQTT	Logging data	Live streaming
External RGB Camera	Camera Based Pose Estimation	OpenCV VideoCapture function	Video feeds	Live streaming, Input from USB device
External RGB Camera	Emotion Recognition	OpenCV VideoCapture function	Video feeds	Live streaming, Input from USB device
IMU/EMG Sensor (Delsys)	Postural Assessment	REST API/gRPC	raw timeseries IMU/EMG	Live streaming
IMU Sensor (XSENS)	Wearable Based Pose Estimation	REST API/gRPC	raw timeseries IMU	Live streaming
Wearable (wristworn)	Emotion Recognition	Device dependent	Biometric sensor data	Continuous data stream
Wearable (wristworn)	Heart Rate Estimation	Device dependent	Biometric sensor data	Continuous data stream
Generic Device	Tokenized Platform	REST API/MQTT	Raw Data	Live Streaming
Information Layer Modules	Tokenized Platform	REST API/MQTT	Processed Data	Depending on the case (periodic)
Cyber Threat Detection	AR APP Pilot 1	MQTT	Threat state (binary) to trigger warning and possibly a description of threat	Continuous data stream

5.3.2 Pilot 2

As described in Section 4.2.2, Pilot 2 focuses on Industry 5.0 and envisions two scenarios: one related to Personal Protective Equipment (PPE) training and practice, and one related to shop floor safety and object tracking. Figure 19 shows the SUN Architecture mapping for Pilot 2.

Pilot 2 offers an AR Application for supporting the training steps for the correct application of PPE to the users. In addition, the AR App allows the task assignment and situational awareness aspect.

The AR Application will run on a standard device exploiting **HOLO XR Streaming** application.

Many contents can be integrated within the VR APP including **3D Objects, 3D Scenes, Avatars, and Rooms**.

Within the Information Layer, the Pilot 2 plan to collect data from **External RGB Cameras** and possibly include some **Stock Assets**. These data will be processed by several modules: **OV Object Detection, Gesture Recognition, and Supporting Decision Making Tool.**

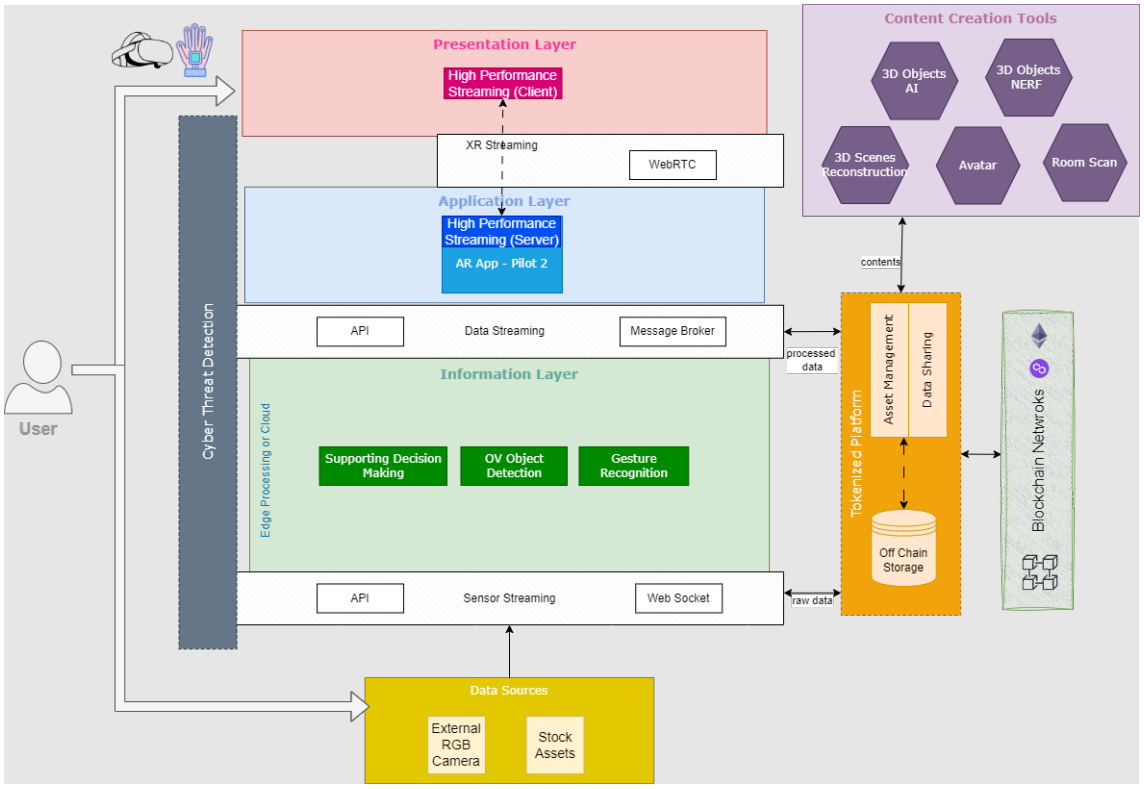

Figure 19: SUN Architecture Mapping for Pilot 2

Table 34 shows all the identified interactions among components within Pilot 2 Architecture.

Table 34: SUN Components Interactions - Pilot 2

Component A (Sender)	Component B (Receiver)	Communication Mechanism	Data Exchanged	Type of Interaction
OV Object Recognition	AR APP Pilot 2	message broker(kafka)	JSON	async (kafka msg)
Supporting Decision Making	AR APP Pilot 2	message broker(kafka)	JSON	async (kafka msg)
Gesture Recognition	AR APP Pilot 2	message broker(kafka)	JSON	async (kafka msg)
AR APP Pilot 2	OV Object Recognition	WebRTC	RGB Images (Frames)	Live streaming
AR APP Pilot 2	Supporting Decision Making	message broker(kafka)	JSON	async (kafka msg)
AR APP Pilot 2	Gesture Recognition	WebRTC	RGB Images (Frames)	Live streaming
AR APP Pilot 2	Client XR (Hololens2 Stream Client)	WebRTC	Frames	Live Streaming

Client XR (Hololens2 Stream Client)	AR APP Pilot 2	WebRTC	User Information	Live Streaming
Room Scan	Tokenized Platform	REST API	Rooms	Sync - on demand
Avatar	Tokenized Platform	REST API	Avatars	Sync - on demand
NERF Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
AI Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
Semantic 3D Scene Reconstruction	Tokenized Platform	REST API	3d scenes	Sync - on demand
Tokenized Platform	AR APP Pilot 2	REST API	Contents	Sync - on demand
AR APP Pilot 2	Tokenized Platform	MQTT or Kafka	Logging data	Live streaming
External RGB Camera	OV Object Recognition	OpenCV VideoCapture function	Video feeds	Live streaming, Input from USB device
External RGB Camera	Gesture Recognition	OpenCV VideoCapture function	Video feeds	Live streaming, Input from USB device
Information Layer Modules	Tokenized Platform	REST API/MQTT	Processed Data	
Cyber Threat Detection	AR APP Pilot 2	MQTT or Kafka	Threat state (binary) to trigger warning and possibly a description of threat	Continuous data stream
Tokenized Platform	Supporting Decision Making	REST API	Stock Assets	Sync - on demand
Tokenized Platform	Gesture Recognition	REST API	Stock Assets	Sync - on demand

5.3.3 Pilot 3

As described in Section 4.2.3, Pilot 3 focuses on two different cases: patients who are recovering from stroke or tetraplegia and patients who are suffering from Apathy.

Pilot 3 offers a **VR Application** for implementing a multi-user virtual space where people can connect, interact, move, and communicate. In the virtual environment, users can communicate and interact via avatars, as well as interact with virtual objects within the space.

In addition, the VR app allows the collection of feedback from the environment and the user and provides it to the user itself via **Thermal Feedback and Brain Stimulation Devices.** The VR Application can run directly on the end-user device or on a standard device exploiting **HOLO XR Streaming** application.

Many contents can be integrated within the VR APP including **3D Objects, 3D Scenes, and Avatar**.

Within the Information Layer, Pilot 3 plans to collect data from **EMG Sensors, External RGB Cameras, IMU Sensors, and Wearables**. These data will be processed by several modules: **OV Object Detection, EMG Decoding, Emotion Recognition, and Multimodal Fusion**.

In Figure 20, the SUN Architecture is applied to the Pilot 3.

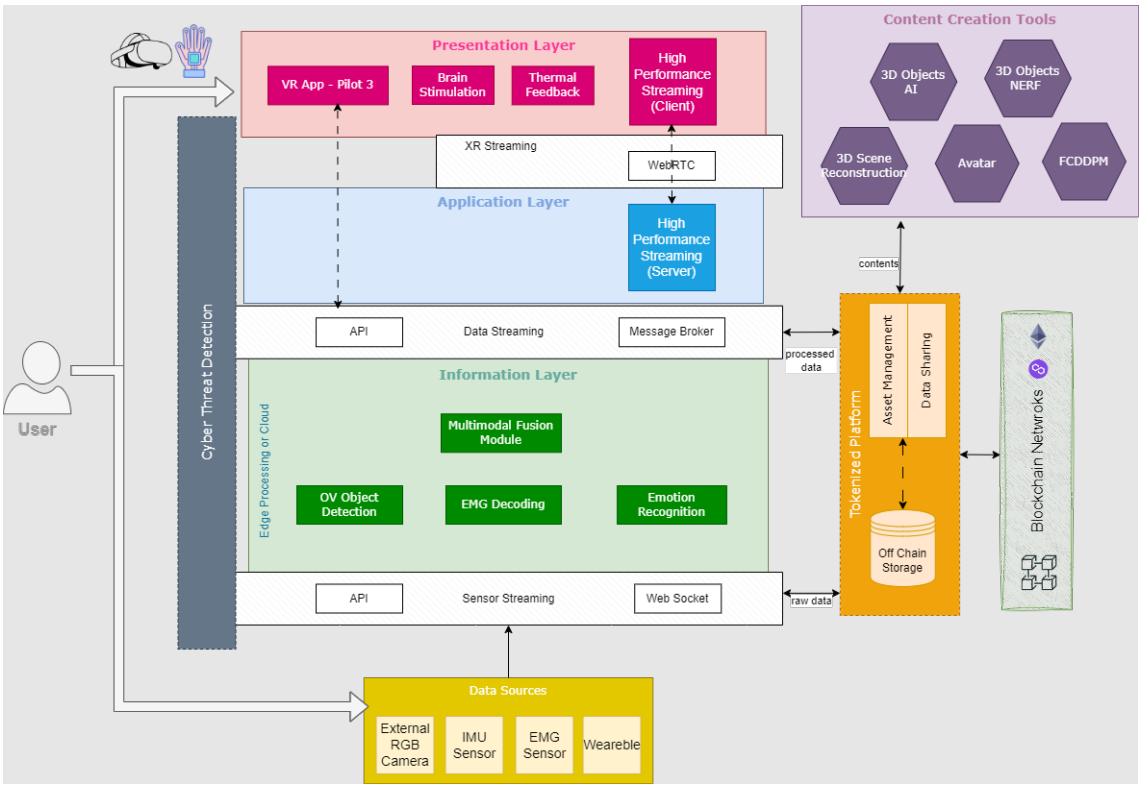

Figure 20: SUN Architecture Mapping for Pilot 3

Table 35 shows all the identified interaction among components within Pilot 3 Architecture.

Table 35: SUN Components Interactions - Pilot 3

Component A (Sender)	Component B (Receiver)	Communication Mechanism	Data Exchanged	Type of Interaction
Emotion Recognition	Multimodal Fusion Module	Function calls	Unimodal emotion recognition results	Data exchange inside monolith app
OV Object Recognition	VR APP Pilot 3	REST API	RGB Images (or streams), JSON strings for results	
Multimodal Fusion Module	VR APP Pilot 3	MQTT/TCP Socket	Human emotions	
EMG Decoding System	VR APP Pilot 3	TCP socket	Decoded commands	Live Streaming
VR APP Pilot 3	Brain Stimulation	TCP socket	Decoded commands (to trigger when necessary)	Live Streaming (only few events)
VR APP Pilot 3	Thermal Feedback	TCP socket	Raw data	Live Streaming
VR APP Pilot 3	XR Streaming Server	WebRTC	Frames	Live Streaming
XR Streaming Server	XR Streaming Client	WebRTC	User Information	Live Streaming
FCDDPM	Tokenized Platform	REST API		Sync - on demand
Avatar	Tokenized Platform	REST API	Avatars	Sync - on demand
NERF Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
AI Reconstruction	Tokenized Platform	REST API	3d objects	Sync - on demand
Acquisition of physical properties	Tokenized Platform	REST API		Sync - on demand
Tokenized Platform	VR APP Pilot 3	REST API	Contents	Sync - on demand

External RGB Camera	OV Object Recognition	OpenCV VideoCapture function	Video feeds	Live streaming, Input from USB device
IMU Sensor	EMG Decoding System	LabStreamingLayer	Raw data	Live Streaming
Wearable	Emotion Recognition	Device dependent	Biometric sensor data	Continuous data stream
EMG Sensor	EMG Decoding System	LabStreamingLayer	Raw data	Live Streaming
Generic Device	Tokenized Platform	REST API/MQTT	Raw Data	Live Streaming
Information Layer Modules	Tokenized Platform	REST API/MQTT	Processed Data	
Cyber Threat Detection	VR APP Pilot 3	MQTT/Kafka	Threat state (binary) to trigger warning and possibly a description of threat	Continuous data stream

6 SUN XR Platform – Technical Specifications

6.1 Architectural Needs

All the activities conducted, and outcomes collected, were important for defining the technical specification of the SUN Architecture, mandatory for the implementation and the integration phase.

In terms of architecture, it is important to support the **communication and integration of the components**, implementing at least the communication mechanisms identified in Section 5.3.

In addition, **cybersecurity requirements** are mandatory due to the type of data and users involved, and the level of maturity of the technologies used, some of these are not advanced enough to be considered a final product.

Finally, was highlighted the importance of having a modular and configurable platform that can be deployed in a distributed manner, including **local deployment, edge infrastructure, and cloud service**.

Details about architectural aspects to be considered during the implementation phase are described in the following paragraphs.

6.2 Communications and Integration aspects

Communication and interoperability aspects are fundamental for ensuring a smooth integration of the SUN components.

As already described, it is important that the SUN Integrated XR Platform support the communication mechanisms and the data exchange among the SUN components at any level. For this reason, three communication layers will be part of the SUN Platform, with different responsibilities and supporting different mechanisms.

In addition, it should be also important to identify which kind of data need to be exchanged within the different layers, to standardize the communication, data format, and data modelling, making the SUN Platform as flexible and adaptable as possible.

Table 36 reports a first list of communication mechanisms identified at the pilot level, together with data exchanged. This list must be intended as not exhaustive, and it could be changed and/or extended during the implementation and integration phase.

Table 36: SUN Architecture Communication Mechanisms

Communication Layer	Protocols and Technologies	Data	Additional Information
Sensor Streaming	Message Broker, Kafka, MQTT, Web Socket, TCP Socket, OpenCV Video Capture, LabStreamingLayer	Sensor Data, Raw Data	Live Streaming, High Frequency
Data Streaming	Message Broker, Kafka, MQTT, REST APIs	Processed Data, XR Contents	Contents OnDemand, Continuous Data Streaming
XR Streaming	WebRTC, TCP Socket	XR Stream,	Live Streaming, High Performance

The integration of components and the release of the SUN XR Platform must follow the specifications and characteristics described in Section 5.

In order to deliver a flexible and effective solution, for the platform releases a **continuous integration and continuous testing** methodology will be adopted, ensuring a high level of quality.

All the technical partners, responsible for the implementation of the different technologies and tools, will be actively involved in the integration phase and will be in charge of ensuring the release of components that reach a sufficient level of technology readiness.

As planned, the **SUN integrated XR platform will be delivered in 2 stages (M26, M32)**. The first version aims to implement key features of a partial workflow, and the second will implement fully functional end-to-end processes, for the complete validation.

Due to the complexity and the number of tools and technologies to be integrated, each technical partner should provide a time-plan for its technology releases as part of the technology maturation toward an end-to-end fully integrated platform.

Following the outcome expected and described in Section 5, at least **3 instances of the platform will be released**, one for each pilot, ensuring the completeness of the functionalities implemented at the pilot level and following the specific deployment requirements for each of them.

6.3 Cybersecurity Aspects

SUN takes cyber security into consideration already from the design stage. It takes a set of complementary measures that provide multifaced security of its data and systems, including their confidentiality, integrity, and availability. They include both preventive and reactive measures.

In brief, SUN includes the preventive security of authentication and applied cryptography for data transfer security and its tokenized platform, as well as reactive security in the form of detection.

The authentication is at the level of the individual pilots' apps, and it aims to prevent the impersonation of users by malicious entities. Authentication is provided in the standard password or PIN-based means that is standard in the industry. The architecture is sufficiently modular to allow for multi-factor authentication in the future for more security-critical areas of application.

Applied cryptography is used at two levels. First, it is used by the communication security protocol employed for the streaming of XR data through the HOLOLIGHT technology. Second, it is used at the level of the digital asset tokenisation of the project, which employs blockchain technology to establish the digital tokens' integrity.

Finally, detection is used in the dedicated XR cyber threat detection tool developed in SUN, which warns the user when a security incident is observed to have been affecting the XR environment.

The following subsections provide more detail on the specifications of the corresponding technical components.

6.3.1 XR Streaming Security Specifications

The XR application can operate either on an on-premises server within an organization's internal network or within a public cloud infrastructure accessible over the Internet. This decision lies with the customers who possess complete authority over determining the location where their sensitive data will be stored securely. Once the process of Remote Rendering is initiated, the data ceases to be stored on the XR device itself and instead is transmitted in real-time. This transition plays a pivotal role in maintaining a high level of security particularly in scenarios involving potential loss of the device or cyber-attacks

by unauthorized individuals. Moreover, ensuring adherence to the prevailing security protocols and standards is facilitated by the streamlined approach of centralized management. Additionally, the architecture of the system dictates that a single server application is capable of transmitting data exclusively to a singular client, thereby rendering any covert interception of the video stream practically impossible without raising any suspicion or being detected by the involved parties.

The Hololight Stream uses the WebRTC protocol which is an open-source project that provides real-time communication (RTC) via simple application programming interfaces. By leveraging the following features, the Hololight Stream SDK provides a robust and secure platform for real-time communication and data transmission. It is open source and supported by major tech companies like Apple, Google, Microsoft, and Mozilla, which means that the technology undergoes regular scrutiny from a broad community of developers, making it less susceptible to hidden vulnerabilities. In fact, it receives regular updates and patches to address any security vulnerabilities or issues that may arise.

More specifically, WebRTC supports encryption for all data transmitted between peers, ensuring that communication remains confidential and secure. It also features Session Control and Signalling Security. This involves exchanging session control information, network data, and media data, which is essential for establishing a connection but is separate from the actual data transmission. By using secure channels for signalling, such as HTTPS, potential vulnerabilities are minimized. WebRTC's Interactive Connectivity Establishment (ICE) protocol ensures that peers can establish connections even in challenging network conditions, while also maintaining security. ICE can utilize STUN or TURN servers to facilitate connection establishment, with TURN providing additional security by relaying traffic if direct peer-to-peer communication is not possible. Additionally, it offers DNS Service Discovery to avoid the need for users to manually input IP addresses, which helps in resolving hostnames to IP addresses automatically. This enhances usability without compromising security. When a direct connection is not feasible or desirable, WebRTC allows the use of a signalling server, which acts as an intermediary. This avoids the direct exposure of IP addresses to potential attackers, enhancing security. Finally, its Transport Agnostic Session Description Protocol (SDP), which describes multimedia sessions and configurations, is used for negotiating media capabilities and transport addresses in a standardized manner. This helps ensure interoperability and reduces the risk of security vulnerabilities in the negotiation process.

6.3.2 Tokenized Platform Security and Data Access Management

Blockchain technology and Smart Contracts can support data sharing without losing control and ownership of it.

Blockchain and Smart Contracts can address three important challenges:

- ensure data privacy,
- manage data access and usage control,
- incentivize secure data sharing.

Through smart contracts, it is possible to track who shared what, with whom, when, by what means, and for what purposes in a verifiable fashion.

The Tokenized Platform exploits the blockchain technology to ensure ownership and digital asset management. Every digital asset transaction, starting from the Content Creation Tools, is recorded on the blockchain, ensuring the immutability of metadata and additional information about the digital asset. Blockchain applies cryptographic hashing for storing metadata and information about the digital asset, ensuring a high level of security about data.

In addition, all the data are stored off-chain, in a standard database, ensuring that all the data access is tracked and monitored through specific Smart contracts. These Smart Contracts will also allow for fine-grained monitoring and management of these actions as well as their translation into tokenized-based values.

6.3.3 Detection of XR cyber threats

Unique to SUN for an XR system, the architecture includes the XR cyber threat component that is designed to react to any failure of SUN's preventive cyber security measures. The component is modular enough to allow for both behaviour-based and knowledge-based detection of threats. Behaviour-based detection relates primarily to deviations of behaviour from what is known as a normal baseline as represented by a set of measurable system characteristics. Knowledge-based detection relates in essence to similarity with the impact characteristics of known threats. By allowing for both behaviour and knowledge-based detection, the architecture allows SUN to be adapting its security to both known and unknown types of threats, especially given the relative immaturity of the field of XR cyber security (i.e., the lack of knowledge of what threats will prove important in the future uses of XR).

Both types of detection are provided in the form of a set of data monitoring tools, an AI pipeline, and an XR visualisation tool that displays the AI's decision in the form of a warning to the user when an XR threat is suspected to have materialised.

6.4 Deployment Aspects

We started investigating the deployment aspects of the three pilots. We focused on gathering additional hardware and software requirements to ensure we're equipped with the necessary PCs for deployment at the pilot sites. For each component, we collected the required operating system for execution during the pilot run, acknowledging that during the offline training and preparation phase, components may operate in varied environments. Additionally, we gathered the GPU RAM requirements.

Most of these requirements are provisional at this stage and they could be updated during the second half of the project. Regarding operating systems, it appears that Windows can support all components during pilot execution, despite the training phase being conducted on a Linux environment.

Regarding GPU RAM, several components necessitate GPUs with RAM ranging from 4 GB to 16 GB. While some of these requirements are yet to be confirmed, we will probably require a PC equipped with several GPUs and plenty of RAM.

7 Conclusions

The SUN Reference Architecture, SUN Components, and related technical specifications will be the base for the implementation of the modular SUN XR Platform, able to be released and configured for satisfying specific pilots and use cases' needs.

As described in this document, the SUN Reference Architecture can be easily mapped with different configurations, taking into account a series of common technical specifications as well as specific pilots and use cases' requirements.

The involvement of the pilot partners in a collaborative way during the design phase, allowed us to consider their feedback in an early stage and prepare the environment to be ready for the validation in an incremental way and with different configurations.

Starting from this outcome, the next steps will focus on the implementation and integration aspects, following the specifications and characteristics described in this document.

The first version of the SUN integrated XR platform will be delivered at M26 in 3 different instances one for each pilot.

References

Tatić, Dušan & Tešić, Bojan. (2017). The application of augmented reality technologies for the improvement of occupational safety in an industrial environment. Computers in Industry. 85. 1-10. 10.1016/j.compind.2016.11.004.

Burova, A., Mäkelä, J., Heinonen, H., Becerril Palma, P., Hakulinen, J., Opas, V., Siltanen, S., Raisamo, R., & Turunen, M. (2022). Asynchronous industrial collaboration: How virtual reality and virtual tools aid the process of maintenance method development and documentation creation. Computers in Industry, 140, 103663. https://doi.org/10.1016/j.compind.2022.103663

Hafidz, I. A. A., et al. (2021). Design of Collaborative WebXR for Medical Learning Platform. In 2021 International Electronics Symposium (IES), Surabaya, Indonesia (pp. 499-504). IEEE. https://doi.org/10.1109/IES53407.2021.9593951

Gagnon Shaigetz, V., Proulx, C., Cabral, A., Choudhury, N., Hewko, M., Kohlenberg, E., Segado, M., Smith, M., & Debergue, P. (2021). An Immersive and Interactive Platform for Cognitive Assessment and Rehabilitation (bWell): Design and Iterative Development Process. JMIR Rehabil Assist Technol, 8(4), e26629. https://doi.org/10.2196/26629

Bao, L., Tran, S. V.-T., Nguyen, T. L., Pham, H. C., Lee, D., & Park, C. (2022). Cross-platform virtual reality for real-time construction safety training using immersive web and industry foundation classes. Automation in Construction, 143, 104565. https://doi.org/10.1016/j.autcon.2022.104565

Schönauer, C., Kaufmann, H., Roussou, M., Rüggeberg, J., Rüggeberg, J., Katsikaris, L., Rogkas, S., & Christopoulos, D. (2023). Creating Informal Learning and First Responder Training XR Experiences with the ImmersiveDeck. Disponibile da arXiv, cs.HC, eprint 2303.04438.

Tümler, J., Toprak, A., Yan, B. (2022). Multi-user Multi-platform xR Collaboration: System and Evaluation. In: Chen, J.Y.C., Fragomeni, G. (eds) Virtual, Augmented and Mixed Reality: Design and Development. HCII 2022. Lecture Notes in Computer Science, vol 13317. Springer, Cham. https://doi.org/10.1007/978-3-031-05939-1_6

Appendix A: Architectural Components Specifications - Template

WP and Task reference	<specify in which WP and specific task the component is connected and developed>
Responsible	<specify responsible partner and main contact person>
Name of Component/Service:	<please write here the name of the architectural element>
Type	<*Hardware* or *Software*>
Functionality	<please write here in free text a short description of the operation of this module/component. A list of functions and operations will be a plus>
Input Connections & Interfaces	<please write the components/tools/source from which it receives input (input dependencies) and mention, if possible, the available connection interfaces e.g. API, sensors, etc.>
Output Connections & Interfaces	<please write the components/tools to which it sends the results (output dependencies) and mention also the available interfaces e.g. API, UI etc.>
Software Requirements/Development Language	<specify any software requirements related to the architectural element, explain the Programming Language that is used during the development of the component>
Hardware Requirements	<specify what hardware requirements are of the module, give specifications about the hardware requirements which are necessary for the best functionality of the component. Please specify device compatibilities and restrictions>
Status of the development of the component	<specify if the component is “already developed” or “partially developed” or “to be developed from scratch”. Starting TRL would be a must>