XR2LEARN

DOCUMENT DELIVERABLE
28/02/2025

D3.4 - XR2Learn enablers (v2)

WP3 – XR Technology PUSH

February 2025

Funded by the European Union

Funded by the European Union under the Grant Agreement 101092851. Views and opinions expressed are however those of the athor(s) only and do not necessarily reflect those of the European Union or European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Author	UM
Work Package	WP3 – XR Technology PUSH
Delivery Date	28.02.2025
Due Date	28.02.2025
Classification	Public

Status of deliverable

Action/role	Name	Date (dd.mm.yyyy)
Submitted by	Ioannis Chatzigiannakis (CNIT)	28.02.2025
Responsible (WP leader)	LS	
Approved by (internal reviewer)	Filisia Melissari and Ioannis Chatzigiannakis	18.02.2025

Revision history

Date (dd.mm.yyyy)	Author(s)	Comments
10.12.2024	UM	Initial template
24.01.2025	LS, SUPSI, UM	Partner contributions
07.02.2025	UM	Edited for internal reviewers
24.02.2025	UM, LS, SUPSI	Applying revisions

28.02.2025	UM	Final version for submission

Author(s) contact information

Name	Organization	E-mail
Enrique Hortal, Annanda Sousa, Bulat Khaertdinov	Maastricht University (UM)	enrique.hortal@maastrichtuniversity.nl

- TABLE OF CONTENTS

- LIST OF FIGURES AND TABLES

– LIST OF ABBREVIATIONS

BA	Beacon application
KPI	Key performance indicator
SSL	Self-supervised learning
WP	Work package
XR	Extended reality
Partners' names and acronyms	
CNIT	CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI
F6S	F6S NETWORK IRELAND LIMITED
MAG	MAGGIOLI SPA
LS	LIGHT AND SHADOWS
SYN	SYNELIXIS SOLUTIONS SA
SUPSI	SCUOLA UNIVERSITARIA PROFESSIONALE DELLA SVIZZERA ITALIANA
UM	UNIVERSITEIT MAASTRICHT
HOU	HELLENIC OPEN UNIVERSITY
EADTU	EUROPEAN ASSOCIATION OF DISTANCE TEACHING UNIVERSITIES
EITM	EIT MANUFACTURING SOUTH SRL

- EXECUTIVE SUMMARY

This deliverable, D3.4 XR2Learn Enablers Version 2, provides an update on the development of innovative tools to facilitate the creation and integration of Extended Reality (XR) learning applications enhanced by affective computing. The proposed enablers serve two main purposes: to reduce the workload involved in developing XR learning applications and to promote the personalization and enhancement of the learning experience in a seamless and effortless manner. In the context of Task 3.2, we have made improvements to the following enablers, which are categorized into three groups:

- **Authoring Tool**: enabler 1, a key component that simplifies the creation of XR applications specifically tailored for educational purposes.
- **Personalization Enablers**: enablers 2-6 and additional enablers are components for adding automatic personalization capability to XR applications.
    - Enablers 2-4: These enablers focus on the development of tools for automatic emotion recognition, i.e., engagement, boredom and frustration, utilizing various input data modalities.
    - Enabler 6: This enabler combines the capabilities of the automatic emotion detection components (enablers 2-5) with contextual information to generate personalized suggestions for adapting learning materials. This adaptation, for example, adjusting the challenge level, aims to make the learning experience more engaging and effective.
    - Command Line Interface: a simplified interface that streamlines access to the functionalities of enablers 2-6.
    - Personalization Dashboard: A web-based graphical interface designed to visualize the functionalities and integration of Personalization Enablers.
- **Data Collection Enabler**: Magic XRoom, this innovative VR software, serves as a tool for collecting data. This data is crucial for evaluating the enablers, as it provides the necessary input for emotion detection algorithms.

Moreover, an additional enabler has been developed:

- *Personalization Enablers Template*: serves as a template, to streamline the creation of new enabler components and facilitate the development process.

Additionally, this document reports on the work performed towards integrating Personalization Enablers and Beacon Applications, including:

- *Personalization Integration Template* creation: Unity example project showcasing how to communicate a Unity (VR) Application with the Personalization Enablers.
- Integration of *Magic XRoom with the Personalization Enablers* showcase.

- *Data Collection Modules*: software plugin currently under development, designed to configure and manage sensors and devices for data collection purposes.

In conclusion, the deliverable presents a set of novel enablers that can be utilized to accelerate the development of educational XR applications. Moreover, by integrating XR applications with the required equipment, data collection modules and emotion recognition enablers, it paves the way for a more immersive, personalized learning experience that is adaptable to the emotional states of the users, thereby enhancing the overall effectiveness of the educational process.

1. INTRODUCTION

This deliverable provides the **progress update** achieved in Task 3.2, "XR2Learn Enablers", during Months 15-26 of the XR2Learn project. This document complements Deliverable 3.2, submitted in Month 14 of the project. Specifically, each section of this document first summarizes the progress reported in the first version and continues with the new developments during the last year.

Deliverable 3.2 presented the initial versions of the six enablers developed by the XR2Learn consortium partners in accordance with the proposal specifications. Additionally, several other enablers were created that were not part of the initial proposal but were developed to support XR educational applications' development, use, integration, and data collection.

The main goals for the second sub-phase of Task 3.2 are to improve the existing enablers and demonstrate the integration between these enablers and Beacon Applications. This focus is also reflected in the content of this report.

Table 1 summarizes XR2Learn enablers and their status as of Month 14 of the project, along with a brief overview of the updates achieved over the past year. Table 2 lists the software delivered in the scope of Task 3.2, along with their GitHub repository links and corresponding licenses.

Table 1. Enablers status: summary and progress

Enabler/tool	Progress	
	Until M14	M15-M26
Enabler 1. Authoring tool Interact	Introduction of INTERACT as a generic tool for creating physics-based VR training scenarios. Demonstration of the practical application of INTERACT as an authoring tool in Beacon Application 1 and Magic XRoom.	Implementation of new features (hand tracking, realistic avatars, cables), performance optimization, and user interface improvements. Technical support provided for Open Call 1 projects using INTERACT.
Enabler 2: Emotion Representation Learning Tools Enabler 3: Tools for Using Emotion Representations Enabler 4: Emotion Classification Tools	Software architecture design and implementation, together with the initial set of models within the training and inference pipelines based on research conducted for audio and bio-measurement modalities. Self-Supervised Learning: operates without the need for labeled data to pre-train Deep Learning models and allows to train models on emotions with less annotated data and resources. Supervised Learning: requires labeled data and provides a structured approach to identify user emotions from input modalities.	Implementation of body-tracking models within the pipelines. Update of inference tools to near real-time processing.
Enabler 5: Multimodal Fusion	Software architecture creation complete. The first version of the implementation did not include all modalities and replicated one modality output prediction.	Implementation of multimodal late fusion for *bio-measurements* and *body tracking* modalities, as well as integration with *emotion classification* and *Personalization tools* in near

		real-time and asynchronous communication.
Enabler 6: Personalization Tool	The initial implementation version included a simplified personalization heuristic to calculate suggestions based on automatically detected user emotions and the challenge level of the activity.	The current implementation version includes a more sophisticated personalization heuristic, which also takes into consideration the user level to calculate suggestions for personalization.
Magic XRoom	Implementation of a VR application for data collection of multimodalities including four scenarios.	Several improvements to usability, graphics, user interface and data collection, changing scenarios to increase the likelihood of eliciting specific engagement states, i.e., boredom and frustration.
Personalization Dashboard	Tool to visualize the inputs and output of Personalization Tool and support integration. Initial release including two versions: version one simulates all the components connected with the Personalization Tool, i.e., the VR application and the output of Inference Tools. Version two simulates the VR application while connecting to and receiving output from Inference Tools.	Release of an additional version to support integration with VR applications, featuring actual integration display instead of simulation. Display of emotional metrics, i.e., boredom, engagement, and frustration percentages.
Command Line Interface	Tool to streamline accessing Personalization Enablers functionalities. Support for audio and bio-measurements modality. No support for multimodality yet.	Support for body tracking modality and multimodality (bio-measurement and body tracking). Support for Personalization Dashboard in near real-time settings included.
(new) Personalization Components Template	It was in the early stage of progress.	Finished and delivered to facilitate extension and creation of new personalization enablers by external contributors.
(new) Personalization Integration Template	It was not implemented.	Unity application delivered to serve as a template for integrating beacon applications with the Personalization Enablers.
(new) Data Collection Module/plugin	It was not implemented.	Extraction into a standalone component is in progress. The virtual reality framework used needs to be converted to the one supported by the BA1 to ensure full integration.

Table 2. List of software delivered with their GitHub repository links and licenses

Software	Repository Link	Open-Source License
Authoring Tool: INTERACT	https://github.com/XR2Learn/enabler-interact	Proprietary software (closed source)
Training Tools	https://github.com/XR2Learn/Personalization-Enablers-Training-Tools	Apache 2.0
Inference Tools	https://github.com/XR2Learn/Personalization-Enablers-Inference-Tools	Apache 2.0
Personalization Tool and Personalization Dashboard	https://github.com/XR2Learn/Personalization-Tool	Apache 2.0
Command Line Interface	https://github.com/XR2Learn/Personalization-Enablers-CLI	Apache 2.0
Personalization Integration Template	https://github.com/XR2Learn/Personalization-Integration-Template	Apache 2.0
Personalization Components Template	https://github.com/XR2Learn/Personalization-Components-Template	Apache 2.0
Magic XRoom	https://github.com/XR2Learn/magic-xroom	MIT
Data Collection Module/Plugin	To be released	MIT

The remainder of the document is organized as follows: we will first report on the progress of the Authoring Tool, followed by updates on the Personalization Enablers and the data acquisition process. Finally, we will discuss the efforts made and the progress achieved in integrating the Personalization Enablers with Beacon Applications.

2. AUTHORING TOOL: ENABLER 1

2.1. INTERACT: AUTHORING TOOL

2.1.1. Authoring Tool Summary up to M14

The previous version of the deliverable introduced INTERACT and its basic functionalities, addressing the need for realistic and quick XR training scenario development. As a Unity plugin, INTERACT enables the creation of physics-based VR training applications, extending beyond basic functionalities to incorporate advanced physics, ergonomic analysis, and interactive scenario design.

INTERACT reduces the technical barrier with its no-code or/and low-code, interface, empowering users to create immersive training solutions for industries like heavy manufacturing and energy. The practical example of integrating INTERACT in XR scenarios was previously introduced in Beacon Application 1, a laser cutting machine training program and later implemented in the MagicXRoom.

The previous deliverable also introduced a set of key features, including a cutting-edge physics engine for realistic interactions, precise collision detection, and intuitive tools for object manipulation. The scenarization module allowed for gamified and pedagogical training experiences, while support for CAD and point cloud data ensured flexibility. INTERACT is proprietary software distributed as a closed source, with the official documentation available at: INTERACT documentation. Figure 1 shows a highlight of INTERACT's key features.

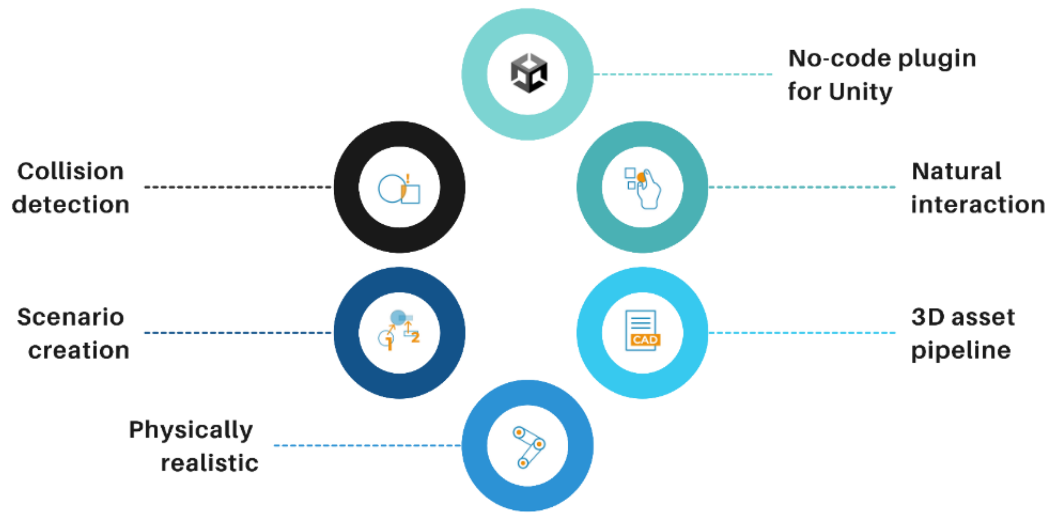


Figure 1. Authoring tool key features

2.1.2. Authoring Tool Updates M15-M26

During the second sub-phase of Task 3.2, the efforts focused on enhancing the INTERACT authoring tool by implementing new features, optimizing performance, and improving the user interface. Additionally, technical support was provided to Open Call 1 (OC1) projects. These advancements further establish INTERACT as a versatile enabler for the XR community.

2.1.2.1. Transition to OpenXR

A major update to the Authoring tool was the transition to the OpenXR standard. SteamVR was replaced with OpenXR, providing a more future-proof and widely supported platform for XR development.

This transition enabled the direct integration of finger-tracking technology within headsets like the MetaQuest 3 and Vive Focus 3, enhancing the natural interaction capabilities of INTERACT. As a result, in just a click, the user can switch from controller interaction to natural hand tracking seamlessly. Together with the embedded physics engine, robust hand-tracking functionality in the authoring tool allows users to interact with virtual environments using natural gestures. This improves immersion and usability for training and simulation scenarios. Figure 2 shows a user interacting in VR with hand movements, demonstrating hand-tracking functionality and increased immersive potential.

OpenXR also unlocked the support of body tracking sensors and glove sensors for reconstructing body and hand postures in INTERACT if needed.

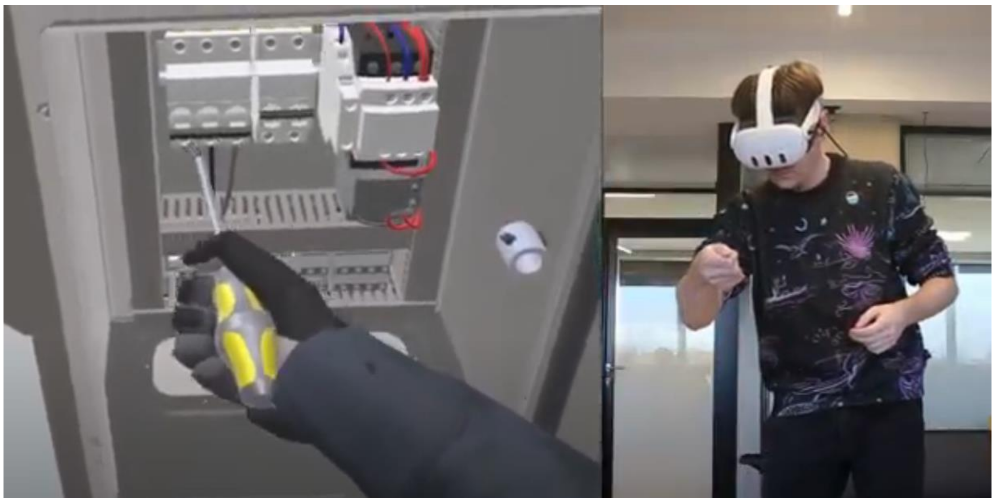


Figure 2. Manipulating a virtual screwdriver with MetaQuest 3 Hand tracking

2.1.2.2. Custom Avatars

Users can now create and customize realistic avatars, making simulations more engaging and tailored to specific training needs. This was made possible by connecting INTERACT with the API from ReadyPlayerMe. Through this integration, users can configure realistic avatars by selecting various attributes such as gender, morphology, skin color, and outfits. This new INTERACT feature not only enhances the realism of simulations but also promotes inclusivity by allowing avatars to better align with diverse scenarios and user demographics. Figure 3 displays some examples of customized avatars.

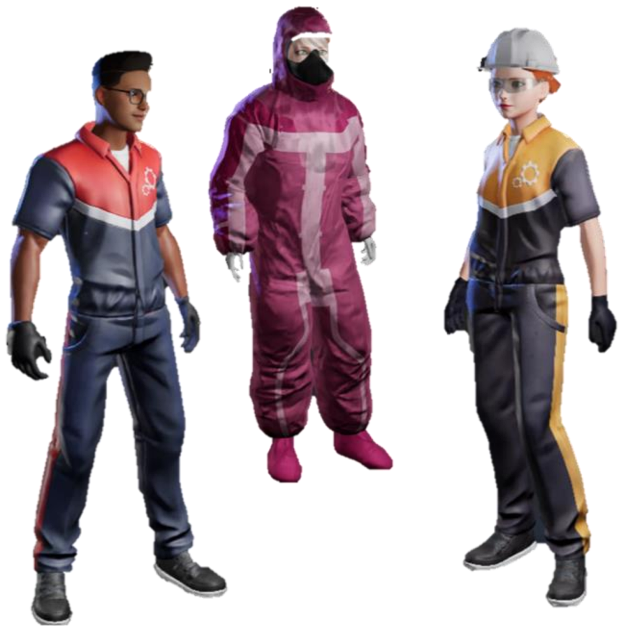


Figure 3. Customizable avatars to the use case

2.1.2.3. Simulation Features

The unique feature provided in INTERACT, namely the physics engine, has been improved and functionalities to better match specific industrial use cases listed below have been added.

- Closed-loop kinematics: This feature allows for more accurate and realistic simulation of mechanical systems with complex kinematic chains (e.g., cranes, industrial robots, etc.)
- Cable: users can now simulate deformable bodies such as cables. The process of creating and configuring a cable has been thought to be quick and intuitive, either from a 3D model or from a spline shape offering designers greater flexibility in crafting intricate cable systems. The ability to dynamically adjust cable lengths during runtime was also introduced, allowing to simulate lifting devices (winch)

  Physics simulation upgrades: Performance and accuracy in physical interactions were also optimized, ensuring smoother and more realistic simulations. This was confirmed by some FSTP projects using INTERACT and witnessing better simulation performance for their use cases.

2.1.2.4. Platform Compatibility Updates

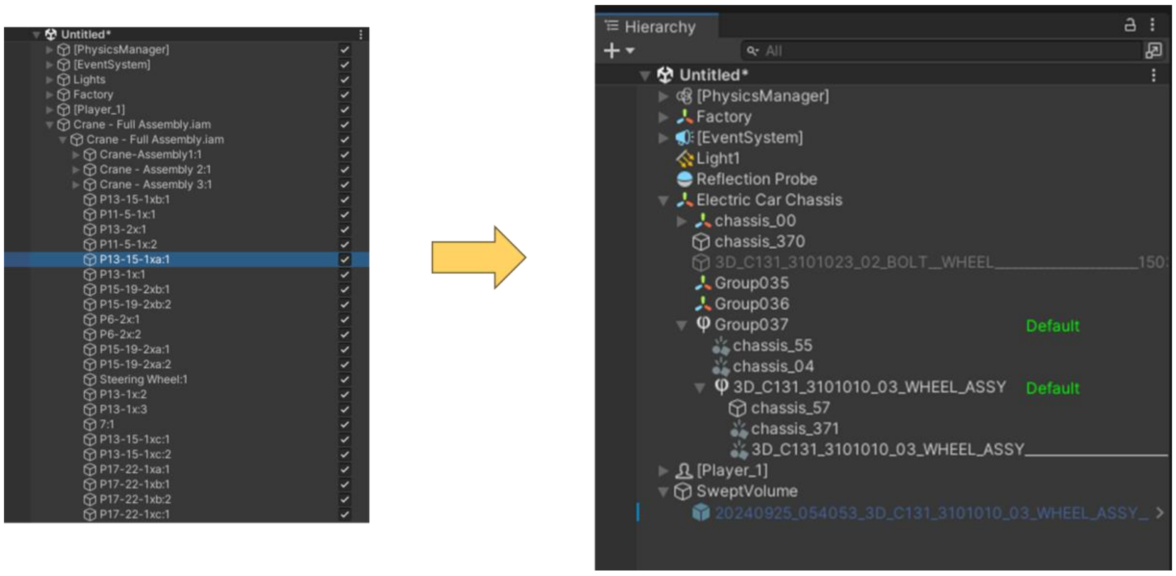
The authoring tool is delivered as a plugin to the 3D engine Unity. Therefore, it is crucial to regularly follow the engine updates and upgrade INTERACT accordingly. We achieved compatibility with Unity 2022+ and Pixyz 2.0, ensuring access to the latest tools and features for developers. In addition, significant efforts were made to keep INTERACT aligned with the regular updates of Unity, including the transition towards Unity 6. Investigating Unity 6 compatibility was a key priority in 2024 when Unity 6 was still experimental. This new Unity version introduces groundbreaking features such as enhanced AI-based tools, improved asset loading performance, and advanced graphics rendering capabilities. These updates are critical for ensuring that INTERACT remains at the forefront of XR development. At the time of writing this document, INTERACT is now fully compatible with Unity 6 and PiXYZ 3.0.

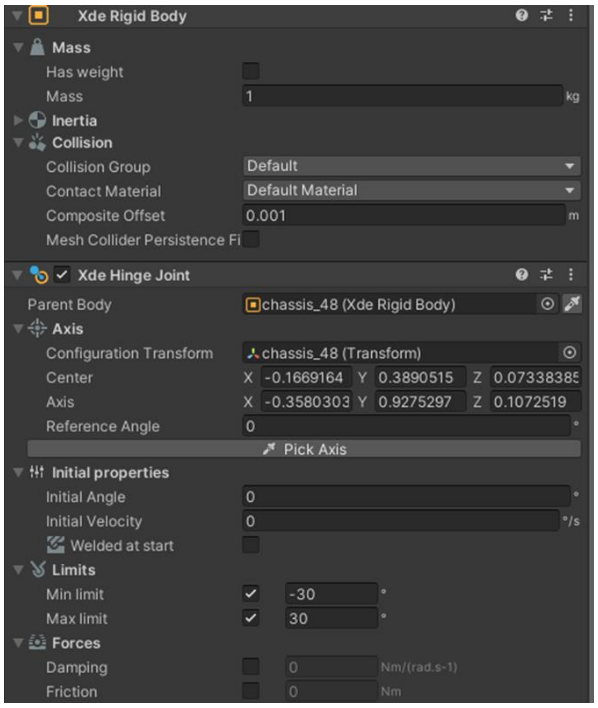
Furthermore, initial explorations began into the feasibility of running INTERACT simulations on native Android headsets, paving the way for broader hardware support and increased accessibility for users. The initial results are encouraging and Android headsets compatibility should be achieved by the end of 2025.

2.1.2.5. Usability Enhancements

In order to streamline the Authoring Process, tools and workflows were revised to significantly reduce the time required to transition from 3D models to fully interactive VR environments. The following achievements were performed during the second phase:

- New user interface features, as shown in Figure 4.
  - Hierarchy icons: Enhanced visual organization, making it easier to navigate and manage scene hierarchies.
  - Units on input fields: Added clear units of measurement to input fields, improving clarity and reducing errors.
  - Foldable sections: Implemented collapsible UI sections to streamline the interface and reduce visual clutter.
  - Undo history: Enhanced undo functionality to allow users to recover from mistakes more effectively.
- An embedded object and material library was created, providing a repository of pre-built assets for users, fostering efficiency and creativity.
- Integrated the In-Editor Tutorials (IET) package to deliver interactive tutorials and tooltips directly within the editor, guiding users through the authoring process step-by-step, as shown in Figure 5.





Xde Rigid Body

Mass
- Has weight: ☐
- Mass: 1 kg

Inertia

Collision
- Collision Group: Default
- Contact Material: Default Material
- Composite Offset: 0.001 m
- Mesh Collider Persistence Fi: ☐

Xde Hinge Joint
- Parent Body: chassis_48 (Xde Rigid Body)

Axis
- Configuration Transform: chassis_48 (Transform)
- Center: X -0.1669164 Y 0.3890515 Z 0.07338385
- Axis: X -0.3580303 Y 0.9275297 Z 0.1072519
- Reference Angle: 0 °

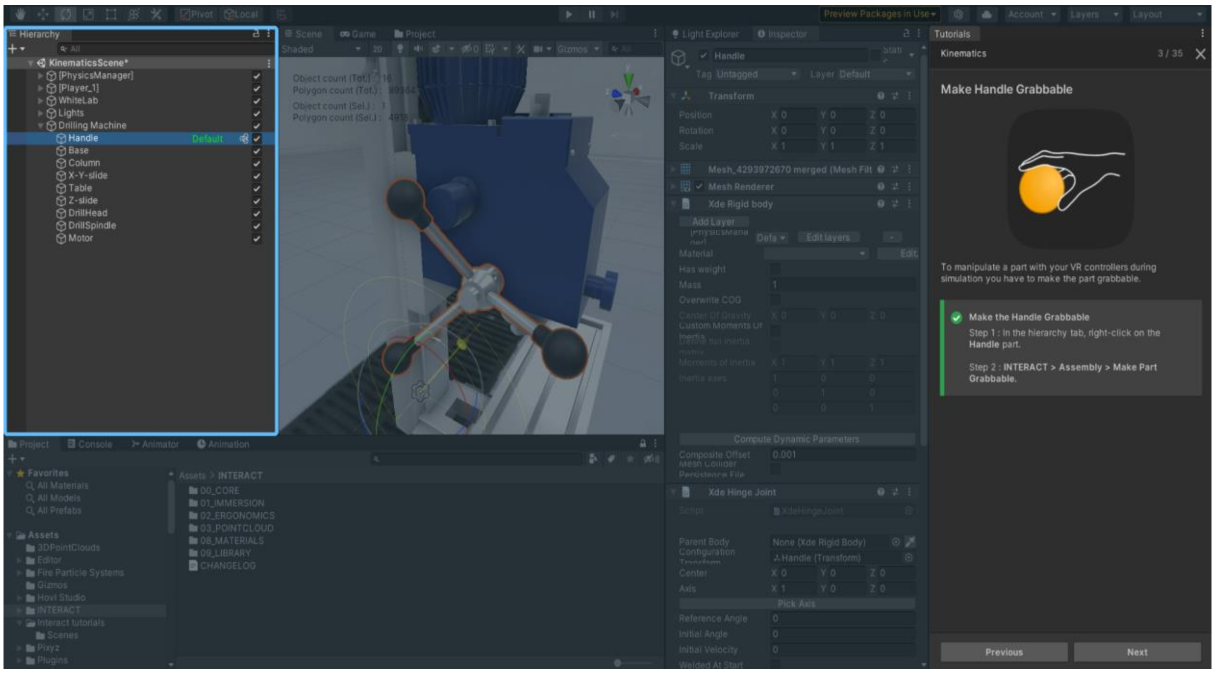
Pick Axis

Initial properties
- Initial Angle: 0 °
- Initial Velocity: 0 °/s
- Welded at start: ☐

Limits
- Min limit: ✓ -30 °
- Max limit: ✓ 30 °

Forces
- Damping: ☐ 0 Nm/(rad.s-1)
- Friction: ☐ 0 Nm

Figure 4. A more comprehensive and intuitive user interface



Tutorials

Kinematics 3 / 35

Make Handle Grabbable

To manipulate a part with your VR controllers during simulation you have to make the part grabbable.

✓ **Make the Handle Grabbable**

Step 1 : In the hierarchy tab, right-click on the **Handle** part.

Step 2 : **INTERACT > Assembly > Make Part Grabbable.**

Previous Next

Figure 5. Using In-Editor tutorials (IET) Unity package to create interactive editor tutorials and tooltips, helping the user to find their way within Unity's dense interface.

2.1.2.6. Feedback from the Community

The implementation of new features, enhancements in usability, and providing support for OC1 projects that used INTERACT in their projects, as reported in Deliverable 3.3, demonstrated the authoring tool's growing maturity and capability as an enabler for the XR community.

Based on feedback from the OC1 projects, several additional improvements were made to facilitate the installation and update processes:

- A one-click installer was developed to simplify deployment.
- INTERACT became compatible with the Unity Package Manager, streamlining integration and updates.
- The package size was reduced to facilitate easier distribution and deployment.
- Centralized package management was introduced for better organization and control.
- A comprehensive API documentation was developed, providing detailed guidance on INTERACT components and libraries, ensuring developers can easily integrate and extend the tool.

3. PERSONALIZATION ENABLERS (ENABLERS 2-6)

3.1. SUMMARY AND UPDATES

3.1.1. Personalization Enablers Summary up to M14

3.1.1.1. Preliminary Research

The previous deliverable explored methodologies for Emotion Recognition (ER) in XR, focusing on modalities beyond facial expressions, such as speech, bio-measurements, and body-tracking. Specifically, the following aspects were elaborated on:

- **AI models.** Suitable Machine and Deep Learning architectures were identified and evaluated. Besides, various feature extraction techniques and Self-Supervised Learning methods were exploited to address ER using speech, bio-measurements, and body-tracking.
- **Open-source data.** Open-source datasets for training and fine-tuning models were assessed, taking into consideration licenses, the effectiveness of emotion elicitation protocols, and dataset compatibility with XR-based use cases.
- **Challenges in XR.** Based on our analysis, we outlined challenging aspects, such as capturing diverse modalities in XR environments, managing computational demands, and collecting annotations for affective states like the Theory of Flow[1].

The preliminary results of this research were presented at a workshop at the 25th International Conference on Mobile Human-Computer Interaction (2023).

3.1.1.2. System Design and Initial Version

Based on the conducted research and careful design of the system, the initial architecture separated six enablers into four standalone modules or tools, as shown in Figure 6. Each of the tools was delivered as a repository in the XR2Learn's project GitHub repository.

[1] Nakamura, Jeanne, and Mihaly Csikszentmihalyi. "The concept of flow." Handbook of positive psychology 89 (2002): 105.

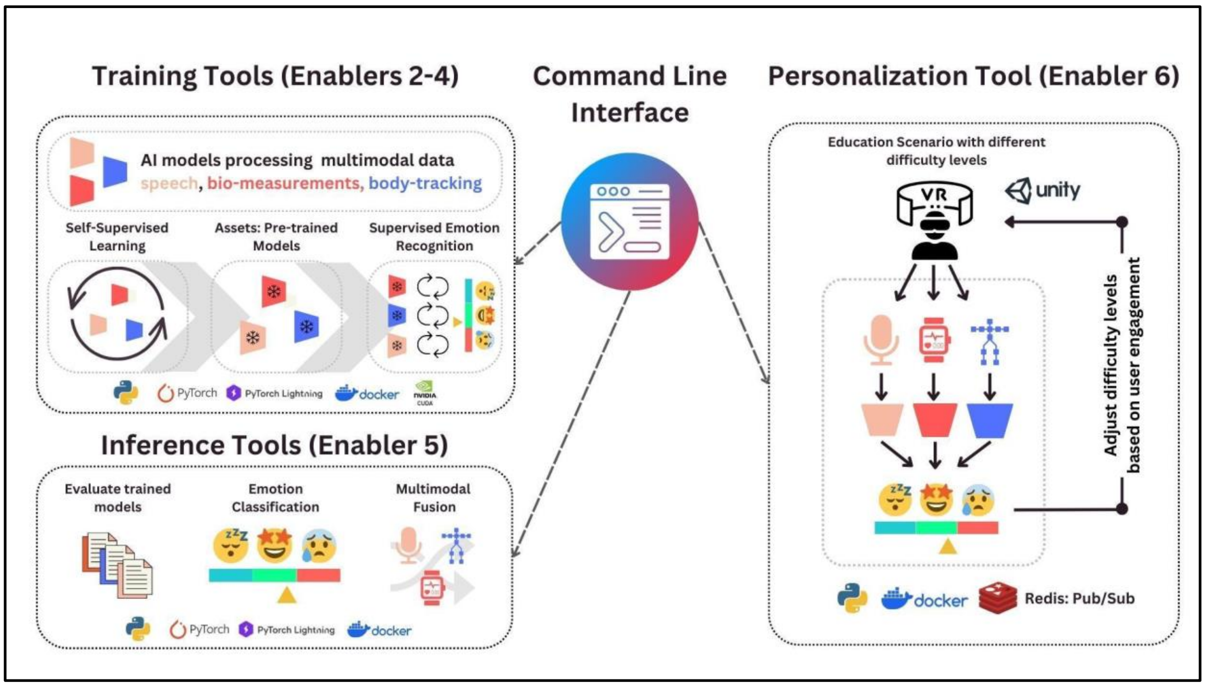

Training Tools (Enablers 2-4)
Command Line Interface
Personalization Tool (Enabler 6)
AI models processing multimodal data
speech, bio-measurements, body-tracking
Self-Supervised Learning
Assets: Pre-trained Models
Supervised Emotion Recognition
PyTorch
PyTorch Lightning
docker
NVIDIA CUDA
Inference Tools (Enabler 5)
Evaluate trained models
Emotion Classification
Multimodal Fusion
Education Scenario with different difficulty levels
VR
unity
Adjust difficulty levels based on user engagement
Redis: Pub/Sub

Figure 6. A high-level overview of the personalization enablers in M14.

In M14, the project delivered the initial versions of the Personalization Enablers. The enablers version at M14 is summarized in Figure 7, according to the supported modalities. The green icon indicates completed implementation, the yellow icon indicates implementation in progress, and the red icon indicates implementation that has yet to start (in M14).

Modality	Dataset	Training Tools	Inference Tools	Personalization Tool	Dashboard	CLI
Audio Signals	Open-Source	✓	✓	✓	✓	✓
Bio Measurements	Project's data collection	✓	✓	✓	✓	✓
Body Tracking	Project's data collection	!	!	✗	✗	✗

Figure 7. Progress in integrating modalities with the personalization enablers' components in M14.

Furthermore, by M14, we identified several improvement points for the Personalization Enablers to be prioritized in the second sub-phase of Task 3.2. Specifically, these include:

- **Body-tracking modality in progress.** As of M14, the body tracking modality was not fully integrated into the Personalization Enablers. While the AI models were researched and implemented, the training and inference pipelines were still not connected. As a result, the multimodal fusion component of the inference tools did not support the combination of body tracking and bio-measurements.
- **Inference tools only supported recorded session data.** As an initial step towards implementing near real-time inference functionality—essential for personalizing VR educational scenarios—the inference tool was firstly designed to support only recorded data, rather than processing data from real-time streams.

- **Showcasing integration.** While the personalization enablers are a set of fully standalone tools, it is important to provide a template application showcasing integration with VR applications using network protocols, to support integration with the Beacon Application(s), open-call project and third-party applications.

3.1.2. Personalization Enablers Update M15-M26

3.1.2.1. Improvements and Integration

During the last year (M15-M26), the emphasis in Personalization Enablers development has been placed on addressing the prioritized points outlined in the previous section and building a foundation for integrating Personalization Enablers with Beacon Applications. The progress achieved, which will be detailed in the following sections, is summarized below. Figure 8 illustrates the current status of the Personalization Enablers in relation to their modalities. An updated diagram of the Personalization Enablers is presented in Figure 9, reflecting the improvements listed below.

- **Personalization Enablers improvements:**
    - **Body-tracking modality.** As of M26, the body-tracking modality is fully integrated into the pipelines of Personalization Enablers, completing KPI 2.3 of the project.
    - **Near real-time inference.** Currently, inference tools support near real-time data processing, which facilitates the exploitation of enablers during education scenarios.
    - **Multimodal Fusion**. Currently, this component combines the prediction of two modalities (bio-measurements and body tracking) in a decision-level approach.
    - **More sophisticated personalization heuristics.** The personalization heuristics have been updated and now include the *user skill level*, in addition to automatically detected *user engagement* and the *activity challenge level*, to calculate personalized suggestions.
    - **Pre-build docker images made available**. This effort aims to streamline the use of Personalization Enablers and promote Open Science principles, particularly reproducibility.
    - **Creation of new enablers.** The Personalization Enabler Template was developed to streamline the process of creating and integrating new enablers with the current software system.
- **Foundations for integration of Personalization Enablers and Beacon applications.** New tools and developments have been created to showcase the communication between different software systems over the network:
    - The *personalization integration template* has been developed.
    - A demonstration of integrating Magic XRoom with Personalization Enablers in near real-time has been performed.
    - *Data* collection module extraction from Magic XRoom is in progress.

Modality	Dataset	Training Tools	Inference Tools	Personalization Tool	Dashboard	CLI
Audio Signals	Open-Source Datasets	✓	✓	✓	✓	✓
Bio Measuments	Magic XRoom	✓	✓	✓	✓	✓
Body Tracking	Magic XRoom	✓	✓	✓	✓	✓

Figure 8. Progress in integrating modalities with the Personalization Enablers' components in M26.

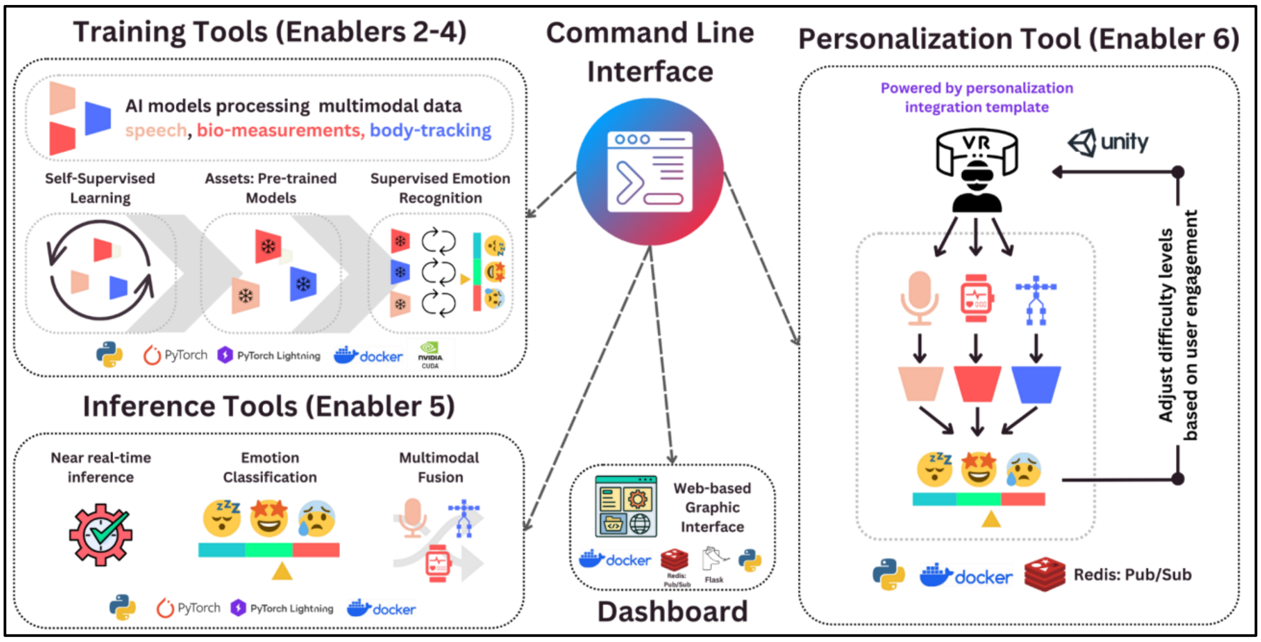

Training Tools (Enablers 2-4)
AI models processing multimodal data
speech, bio-measurements, body-tracking
Self-Supervised Learning
Assets: Pre-trained Models
Supervised Emotion Recognition
PyTorch
PyTorch Lightning
docker
NVIDIA CUDA
Inference Tools (Enabler 5)
Near real-time inference
Emotion Classification
Multimodal Fusion
PyTorch
PyTorch Lightning
docker
Command Line Interface
Web-based Graphic Interface
docker
Redis: Pub/Sub
Flask
Dashboard
Personalization Tool (Enabler 6)
Powered by personalization integration template
VR
unity
Adjust difficulty levels based on user engagement
docker
Redis: Pub/Sub

Figure 9. A high-level overview of the Personalization Enablers in M26, with body tracking fully integrated, evaluation component eliminated from Inference Tools, which now implements a near real-time inference. The Personalization Dashboard is included in the diagram.

The following sections of this chapter provide a more detailed description of new functionalities and updates within each tool of the Personalization Enablers.

3.2. TRAINING TOOLS (ENABLERS 2-4)

3.2.1. Training Tools Summary up to M14

By M14 of the project, the partners focused on the development of the Training Tools, consisting of five key components: Pre-processing, Handcrafted Features Extraction, Self-Supervised Learning (SSL) Training, SSL Features Extraction, and Supervised Learning Training. These components align with Enablers 2, 3, and 4, providing a modularized and flexible framework for pre-training and fine-tuning models as part of the Personalization Enablers.

The architecture is designed to isolate components and dependencies and enable the independent deployment of each modality, such as audio and bio-measurements. During the first 14 months, models for two modalities, audio and bio-measurements, were integrated into the Training Tools pipelines. The workflow begins with the Pre-processing module, which organizes raw data into structured time windows with corresponding labels. The Handcrafted Features Extraction component provides an alternative to Deep Learning-based feature extraction. These features or raw data can be utilized for SSL pre-training or as input for supervised learning tasks. The SSL Training module focuses on unsupervised pre-training of encoders (Enabler 2), enabling the generation of robust feature representations. The Features Extraction module leverages trained encoders to produce embeddings (Enabler 3), while the Supervised Learning Training module fine-tunes these models or trains the model from scratch using labeled data (Enabler 4).

Each component operates independently while being integrated with others within the pipeline. Subsequent training and inference components can seamlessly use the outputs of the components. The audio modality was implemented using the open-source RAVDESS data format, and the bio-measurements modality was implemented using the data format of the XR2Learn data collection tool, Magic XRoom.

3.2.2. Training Tools Updates M15-M26

Within M15-26 of the project, the following major contributions have been made to the Training Tools pipeline:

- **Body-tracking modality**. The body-tracking analysis is supported in the latest versions of the tools. Specifically, functionalities such as pre-processing, hand-crafted feature extraction and supervised training can be applied to body-tracking data collected using the Magic XRoom data format. The pre-processing component segments input time series into shorter time windows, with the length configurable via a configuration file. The supervised component contains a training pipeline for a model, namely a Convolutional Neural Network, trained on the segments obtained to predict users' engagement, according to the theory of flow labels.
- **Discretizing continuous values**. Starting from version 1.3 of Magic XRoom, users provide feedback on a continuous scale (see Figure 19). The pre-processing pipelines have been adjusted to discretize these values into the provided categories, where the threshold values for each category can be set through a configuration file. Later, it might also be interesting to allow users to fit regression models (using the same encoders but different output layers for the neural network) in order to train models to predict the continuous values of engagement.
- **Pre-built Docker containers.** To make the Training Tools components easier to use, all Docker containers have been pre-built and hosted on GitHub. As a result, users do not need to build the Docker containers themselves. Instead, they can simply pull the images, which is faster and reduces the likelihood of bugs or errors. Training Tools encompassed a total of 14 docker images[2] at the time of writing this document.
- The **documentation** has been expanded to include some previously uncovered sections, as well as the new changes that have been made.

[2] https://github.com/orgs/XR2Learn/packages?repo_name=Personalization-Enablers-Training-Tools

Table 3 lists the versions released of Training Tools during M15-M26. The full list of corresponding changes can be accessed in the Training Tools changelog via the following link:

https://github.com/XR2Learn/Personalization-Enablers-Training-Tools/blob/master/CHANGELOG.md

Table 3. Training tools released versions and dates.

Version	Release Date
v0.4.0	2024-03-12
v0.5.0	2024-04-09
v0.6.0	2024-07-30
v0.6.1	2024-08-20
v1.0.0	2024-10-25
v1.1.0	2025-01-13
v1.2.0	2025-01-21

All the released versions can be accessed and downloaded at: https://github.com/XR2Learn/Personalization-Enablers-Training-Tools/releases

3.3. INFERENCE TOOLS (ENABLER 5)

3.3.1. Inference Tools Summary up to M14

The initial prototype of Inference Tools was designed to support unimodal and multimodal emotion classification. The toolbox consists of *emotion classification*, *multimodal fusion*, and *evaluation*, delivered as modularized and standalone components via Docker. The first version of the deliverable introduced these components and provided minimal technical documentation for their usage.

Up to M14, the *emotion classification* component could operate with two individual modalities, namely audio and bio-measurements, to predict emotions using models trained within Training Tools. These modalities were implemented into separated components, the *emotion classification* component for audio modality was developed using the data format of an open-source dataset called RAVDESS, while the emotion classification component for bio-measurements modality was developed using the data format of Magic XRoom.

The initially released *multimodal fusion component* provided a non-real-time pipeline for combining the outputs from individual modalities using a decision-level fusion schema for a series of predictions, implemented as majority voting at the class probability level. Because only one modality using Magic XRoom data format had been fully implemented at this point, i.e., bio-measurements, in practice, the *multimodal fusion component* was simply reproducing the output of this modality, calculated by the *emotion classification component*.

The *Evaluation component* assessed the performance of these emotion detection systems, providing metrics like accuracy, recall, and precision.

These components were designed to operate independently or as part of an integrated pipeline, exploiting the outputs from the Training Tools components, e.g. trained models and pre-processed data. The system supported flexible configurations, enabling users to run components locally or via Docker. Additionally, the fusion tool supported a publisher-subscriber messaging protocol to pass model predictions to personalization modules over a network.

3.3.2. Inference Tools Updates M15-M26

Within M15-26 of the project, the following major contributions have been made to the Training Tools pipeline, and Figure 10 shows the updated Inference Tools components:

- **Body-tracking modality.** First, during M15-26, the body-tracking modality has been integrated into all components of inference tools. As a result, the body-tracking models trained within Training Tools can be used in inference.
- **Near real-time pre-processing.** An important contribution in M15-26 is a novel pre-processing component within Inference Tools that allows near real-time processing of data being written by VR applications. Specifically, this component reads sensory data from logs, applies required pre-processing and publishes pre-processed data via Pub/Sub. The current version of the component supports the data format recorded by Magic XRoom. The technical details regarding this component are presented in Section 3.3.2.1.
- **The evaluation component was removed.** After analyzing the possible use cases, the evaluation component was removed from the inference tools in version 1.0.0 due to its redundancy. Specifically, the components in inference tools are used to provide predictions close to real-time for data streaming from a VR application. In turn, model evaluation is normally conducted during model training at the experimentation stage within the supervised training component from Training Tools.
- **Multimodal Fusion**. The multimodal fusion pipeline has been significantly upgraded to support a near real-time inference pipeline. Specifically, the predictions from different modalities are accumulated before being fed to the multimodal fusion logic. This pipeline is elaborated on in Section 3.3.2.2.
- **Extending publisher-subscriber protocol.** In the latest version of Inference Tools, all components can communicate with each other using the Pub/Sub protocol. Specifically, the components publish and/or subscribe to topics dedicated to different pipeline parts, from pre-processed data to final predictions fused across modalities.
- **Pre-built Docker containers.** All Docker containers have been pre-built and hosted on GitHub to facilitate using the Inference Tools components. As a result, users do not need to build the Docker containers themselves. Instead, they can simply pull the images, which is faster and reduces the likelihood of bugs or errors. Training Tools encompass a total of 7 docker images[3] at the time of writing this document.
- The **documentation** has been expanded to include some previously uncovered sections, as well as the new changes that have been made.

[3] https://github.com/orgs/XR2Learn/packages?repo_name=Personalization-Enablers-Inference-Tools

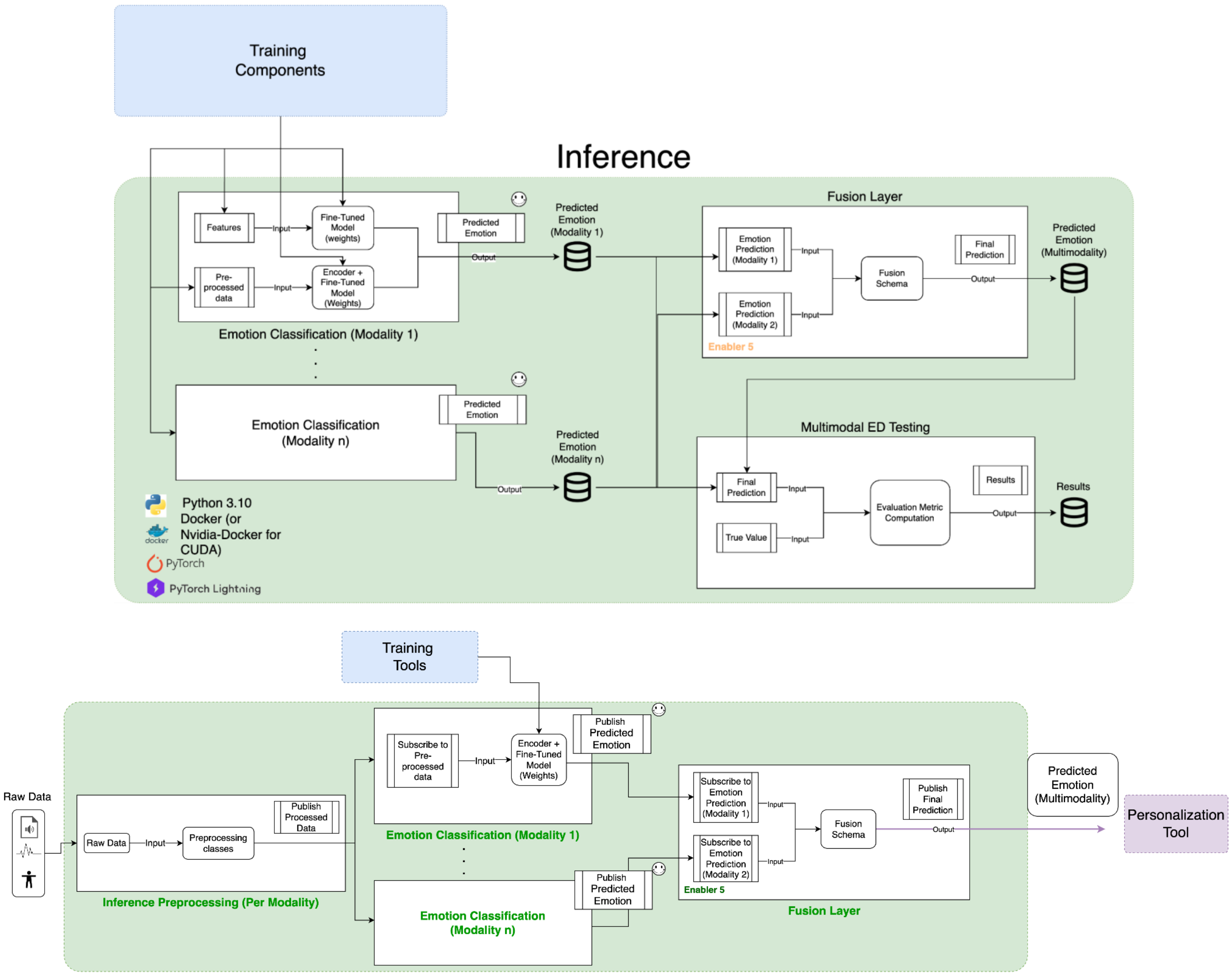

Training Components
Inference
Features
Input
Fine-Tuned Model (weights)
Pre-processed data
Encoder + Fine-Tuned Model (Weights)
Predicted Emotion
Output
Predicted Emotion (Modality 1)
Emotion Classification (Modality 1)
Emotion Classification (Modality n)
Predicted Emotion (Modality n)
Fusion Layer
Emotion Prediction (Modality 1)
Emotion Prediction (Modality 2)
Fusion Schema
Final Prediction
Enabler 5
Predicted Emotion (Multimodality)
Multimodal ED Testing
Final Prediction
True Value
Evaluation Metric Computation
Results
Python 3.10
Docker (or Nvidia-Docker for CUDA)
PyTorch
PyTorch Lightning
Training Tools
Raw Data
Preprocessing classes
Publish Processed Data
Inference Preprocessing (Per Modality)
Subscribe to Pre-processed data
Encoder + Fine-Tuned Model (Weights)
Publish Predicted Emotion
Emotion Classification (Modality 1)
Emotion Classification (Modality n)
Subscribe to Emotion Prediction (Modality 1)
Subscribe to Emotion Prediction (Modality 2)
Fusion Schema
Publish Final Prediction
Enabler 5
Fusion Layer
Predicted Emotion (Multimodality)
Personalization Tool

Figure 10. Inference Tools components state in M14 (top) and Inference Tools components state in M26 (bottom). Before, the communication between components was performed by writing and reading stored files with data. Currently, all components exchange messages in near real-time using the Pub/Sub protocol for asynchronous communication. Also, the current Inference Tools version has its Preprocessing component, which allows near real-time inference latency.

Table 4 lists the versions released during M15-M26, and the full list of corresponding changes for each version can be accessed in the Inference Tools changelog via the following link:

https://github.com/XR2Learn/Personalization-Enablers-Inference-Tools/blob/master/CHANGELOG.md

Table 4. Inference tools released versions and dates.

Version	Release Date
v0.4.0	2024-03 -12
v0.5.0	2024-04-09
v0.6.0	2024-07-24
v1.0.0	2024-10-25

All the released versions can be accessed and downloaded at:
https://github.com/XR2Learn/Personalization-Enablers-Inference-Tools/releases

3.3.2.1. Inference Data Processing Component

Description

A new component in the Inference Tools was created to support the near real-time processing of user data into automatic classification of user engagement levels according to the theory of flow, i.e., boredom, engagement and anxiety.

This component reads the user data that was collected from the VR device and sensors and written into CSV files, pre-processes the files into time windows data according to the frequency of the data modality, and publishes each time window data to be further processed by the other Inference tool components, i.e., Emotion Detection and Fusion Layer components. This component runs in low latency, i.e., near real-time. This component is deployed per modality, i.e., each modality has its own Inference Preprocessing component.

The Inference Data Processing Component behaves similarly to the Preprocessing component from Training Tools (Section 3.2). Both of them read raw data from different modalities and process it into a format to be used by the following components in the systems pipeline. They differ in how they make available their output. While Preprocessing from Training Tools saves the output into files that will be used for pre-training and fine-tuning ML models, the Inference Preprocessing sends the processed data as a message using the Pub/Sub protocol so as to allow the asynchronous and near real-time latency of this communication, a requirement for the Inference components.

Prerequisites

- Docker[4] installed
- Python 3.10[5] installed

Installation

Note: This component must be installed on the same machine as the one running the VR application that records the user's data, as the Inference Preprocessing reads the local files generated by the Unity VR application. For example, if this component is

[4] https://docs.docker.com/engine/install/
[5] https://www.python.org/downloads/

running with Magic XRoom, this component needs to be installed on the same computer as Magic XRoom is running.

1. Download the code at the GitHub repository (See Table 2)

Basic User Manual

1. Navigate to the folder:

"*Inference_Data_Processing -> Inference_Data_Processing_<MODALITY>_Modality*", changing <MODALITY> for the modality name, for example, use *BM* for the bio-measurements modality.

2. Edit the "configuration.json" file to set up the location to read the raw data under the "*<MODALITY> -> inference_config -> data_processing -> monitor_directory*" key. Changing *<MODALITY>* for the used modality name, e.g., "*body_tracking*" for the body tracking modality.
3. Run the docker container by running the command (which will pull the docker image and start the Inference Preprocessing docker service for a given modality):
   a. *docker compose up*

3.3.2.2. Multimodal Fusion Component

As outlined in Section 3.3.1, in the initial version of Deliverable 3.2, the Multimodal Fusion component was still in its early stages. The architecture had been established and connected to the other Inference Tools components, but its functionality was incomplete. It did not support multiple modalities and simply repeated the predictions received from the emotion classification component, acting as a wrapper for posterior developments.

The Multimodal Fusion component now combines multiple modalities for engagement predictions into a single multimodal prediction using Decision Level Fusion, also known as Late Fusion. In this approach, each modality has its own emotion classification model that predicts an engagement-related state. The final Multimodal Fusion prediction is generated after considering the predictions from each individual modality, as can be observed in Figure 10, which depicts the updated diagram of Inference Tools components. By default, the current version of Multimodal Components is set to work with modalities and the data format available in Magic XRoom (Section 4.1), including bio-measurements and body-tracking modalities.

One of the main challenges in multimodal prediction fusion is the temporal synchronization of the data. Different modalities often generate data at varying frequencies, and their machine-learning classifier pipelines produce predictions using specific temporal window sizes based on the nature of each modality. For the Magic XRoom data format, the bio-measurement modality generates data in a 5-second time window after preprocessing, while the body tracking modality produces data every second. As a result, the combined prediction must be generated every 5 seconds to ensure that the fusion component has at least one prediction from each modality during that time window.

The Multimodal Fusion component monitors each modality channel for new messages containing emotion predictions sent asynchronously by the emotion classification components. As a result, the Multimodal Fusion component is then responsible for organizing those messages in a temporal manner to create compatible time windows, allowing this component to combine the predictions into a multimodal output prediction.

When creating the fusion for bio-measurements and body tracking from Magic XRoom data format, the Multimodal Fusion Component creates and manages a buffer to store the messages from both modalities until compatible time windows are completed. This generally means storing five messages from body tracking while waiting for one message from bio-measurements to execute the fusion. This number of messages stored in the buffer might differ if any delays in receiving messages from emotion classification occur. In summary, the Multimodal Fusion Component receives messages from unimodal classification models and keeps them until it is possible to execute a multimodal fusion based on the frequency of each modality data generation.

Figure 11 shows the fusion process for the modalities of bio-measurement and body-tracking. For each time window, the bio-measurement modality produces a single prediction vector of dimension three (representing the probability of each class), while the body-tracking modality generates five prediction vectors (one per time slot), each also of dimension threeFirstly, for a given time window, we need to represent the body-tracking modality with one prediction vector; for this, we use a majority vote heuristic, selecting the first prediction vector for the most present class.

After obtaining one prediction vector per modality for each time window, the two modalities are combined by performing a vector average operation with the prediction vectors from each modality.

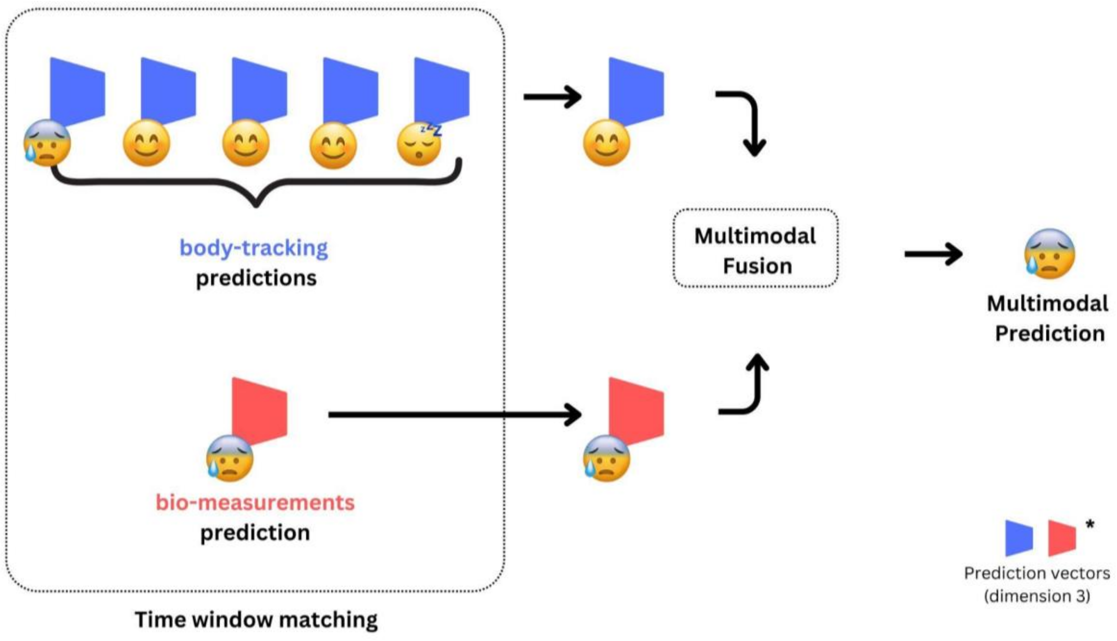


Figure 11. Multimodal Fusion for bio-measurement and body tracking modalities (selection of modalities is based on the two modalities from the Magic XRoom data format fully integrated into the Personalization Enablers).

It is important to highlight that the output of the Inference Tools reflects the user's **engagement state**, which can be *engagement*, *boredom* or *anxiety*. The Personalization Tool utilizes real-time engagement data to calculate suggestions for adjusting educational content when the user is not engaged, specifically when experiencing boredom or anxiety. No further emotional analysis is performed, and no emotional data is stored. Furthermore, the data used and generated by the Inference Tools is non-identifiable. Additionally, the Personalization Tool does not use single emotional states but aggregates a user's engagement states throughout a specific educational VR activity. All of these elements were designed to ensure users' privacy and consider current EU AI Act discussions[6].

3.4. PERSONALIZATION TOOL (ENABLER 6)

3.4.1. Personalization Tool Summary up to M14

Personalization Tool was designed to enhance educational XR applications by providing personalized activity-level recommendations. This tool integrates user engagement predictions from the Inference components with contextual data from VR applications, particularly activity difficulty, to dynamically adjust the difficulty of learning materials. By leveraging real-time data on user emotions and contextual information, the tool makes recommendations that allow delivering educational content at a level optimal for effective and engaging learning.

The tool was implemented to utilize the Pub/Sub messaging protocol to communicate with Inference Tools and VR applications. Besides, the component was integrated with the Personalization Dashboard (Section 3.5.1), which was used to visualize the outputs of inference and personalization components.

3.4.2. Personalization Tool Updates M15-M26

The major update to the Personalization Tool involved enhancing the personalization heuristic. In the previous version, the calculation was based on the *user's emotions,* which are automatically identified by Inference Tools, and *the difficulty level* of activities, as determined by a VR application. With this update, a new metric has been added: *the user's skill level*. This addition completes **KPI 2.4**, as outlined in the project's proposal agreement document.

The emotion representation model utilized, i.e., the emotions considered and the personalized suggestions to the educational scenarios, is based on the Theory of Flow (Figure 12), which maps activity levels (challenges) and user skill levels (action capability) into three emotional/mental states: anxiety, flow and boredom. According to the Theory of Flow, properly adjusting the difficulty level of an activity to match the user's skill can promote and/or maintain a state of flow or engagement. The Theory of Flow has been frequently used in adaptive learning[7] and gaming challenge adaptation[8].

[6] https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng, https://artificialintelligenceact.eu/ai-act-explorer/

[7] Basawapatna, Ashok & Repenning, Alexander & Koh, Kyu Han & Nickerson, Hilarie. (2013). The Zones of Proximal flow: Guiding students through a space of computational thinking skills and challenges. ICER 2013 - Proceedings of the 2013 ACM Conference on International Computing Education Research. 67-74. 10.1145/2493394.2493404.

[8] Chanel, Guillaume & Rebetez, Cyril & Bétrancourt, Mireille & Pun, Thierry. (2008). Boredom, Engagement and Anxiety as Indicators for Adaptation to Difficulty in Games. 13-17. 10.1145/1457199.1457203.

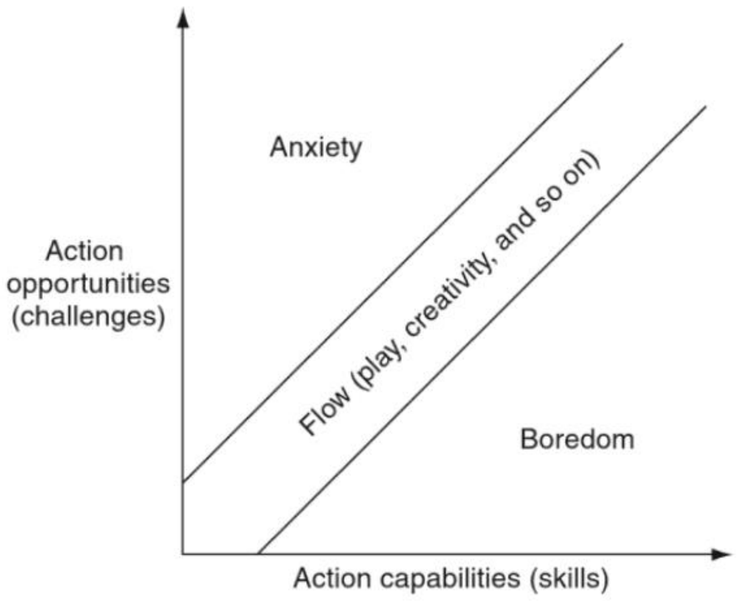


Figure 12. Theory of Flow[9]

The Personalization Tool then has three inputs: the current *activity level* (e.g., low, medium, hard), the *user skill level* (e.g., beginner, intermediate, advanced), and the automatically identified *user engagement state* (e.g., anxiety, engagement, and boredom). Figure 13 shows an overview of the Personalization Tool's inputs and outputs.

Utilizing these three inputs, the Personalization Tool calculates and suggests the next activity level, which could remain the same, increase or decrease the challenge level.

Since the Personalization Tool aims to foster or maintain the engagement (flow state), if it receives the information that the user's emotional state is engaged, the personalized suggestion is that the activity level should remain the same as before because the user is already in the target engagement state.

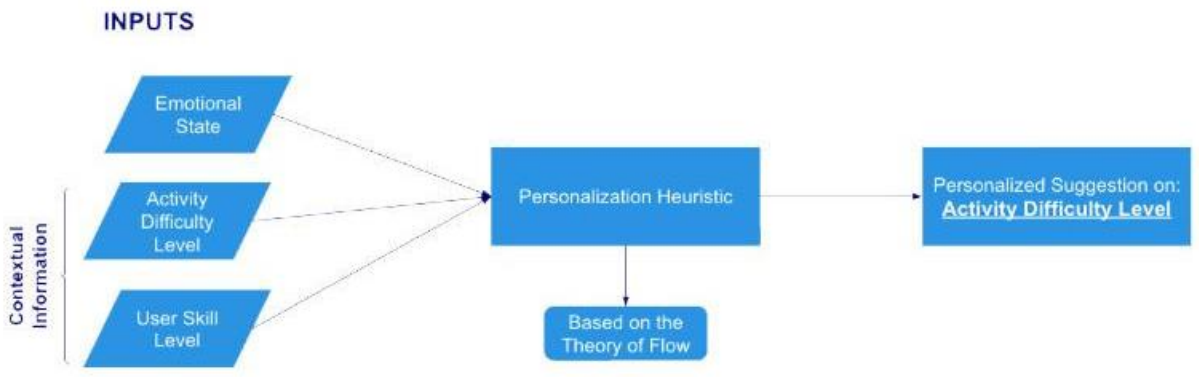

INPUTS
Emotional State
Activity Difficulty Level
User Skill Level
Contextual Information
Personalization Heuristic
Based on the Theory of Flow
Personalized Suggestion on: Activity Difficulty Level

Figure 13. Personalization Tool overview: inputs and output

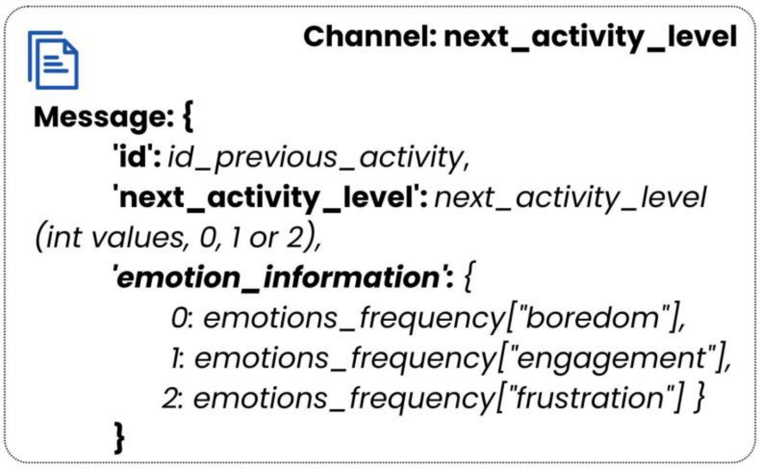
[9] Nakamura, Jeanne, and Mihaly Csikszentmihalyi. "The concept of flow." Handbook of positive psychology 89 (2002): 105.

However, if the Personalization Tool receives information that the user is in an anxiety or boredom state, it will suggest a change in the activity level to bring the user back to engagement.

The personalized suggestion is made two-fold: first, considering the engagement state, and second, the user level. The emotional state will dictate if the activity level will decrease or increase. For instance, if the emotion is anxiety, the activity level should decrease, whereas if the emotion is boredom, the activity level should increase. The user level then dictates how much the activity level should increase or decrease. For example, suppose a beginner user is anxious while doing a hard activity. In that case, the Personalization Tool will suggest the activity level be changed to an easy level, not to a medium level, since the user is a beginner.

Additionally, the current version of the Personalization Tool has an updated message format. The new format now includes the final user's engagement level, reflecting their emotional response during the activity and the percentage of engagement states experienced by the user throughout the learning activity. This change was implemented to provide VR applications with more relevant information, allowing them to integrate personalized suggestions tailored to the activity.

Figure 14 represents the current message Personalization Tools generates.


Channel: next_activity_level

```
Message: {
        'id': id_previous_activity,
        'next_activity_level': next_activity_level
(int values, 0, 1 or 2),
        'emotion_information': {
              0: emotions_frequency["boredom"],
              1: emotions_frequency["engagement"],
              2: emotions_frequency["frustration"] }
        }
```

Figure 14. Message format the Personalization tool generates, including emotion information and suggestion of the next activity level.

Lastly, to make the use of the Personalization tool easier, all pre-built Docker images are hosted on GitHub. This means users do not need to build the Docker containers themselves; they can simply pull the images. This approach is faster and reduces the likelihood of bugs or errors. Two Docker images[10] are included in the Personalization Tool at the time of this writing.

Table 5 lists the versions released during M15-M26, and the full list of corresponding changes for each version can be accessed in the Personalization Tool changelog via the following link:

https://github.com/XR2Learn/Personalization-Tool/blob/master/CHANGELOG.md

[10] https://github.com/orgs/XR2Learn/packages?repo_name=Enabler-6-Personalisation-Tool

Table 5. Personalization Tool released versions and dates.

Version	Release Date
v0.2.2	**2024-03-12**
v0.2.3	**2024-07-25**
v1.0.0	**2024-10-28**

All the released versions can be accessed and downloaded at: https://github.com/XR2Learn/Personalization-Tool/releases

3.5. ADDITIONAL ENABLERS

3.5.1. Personalization Dashboard

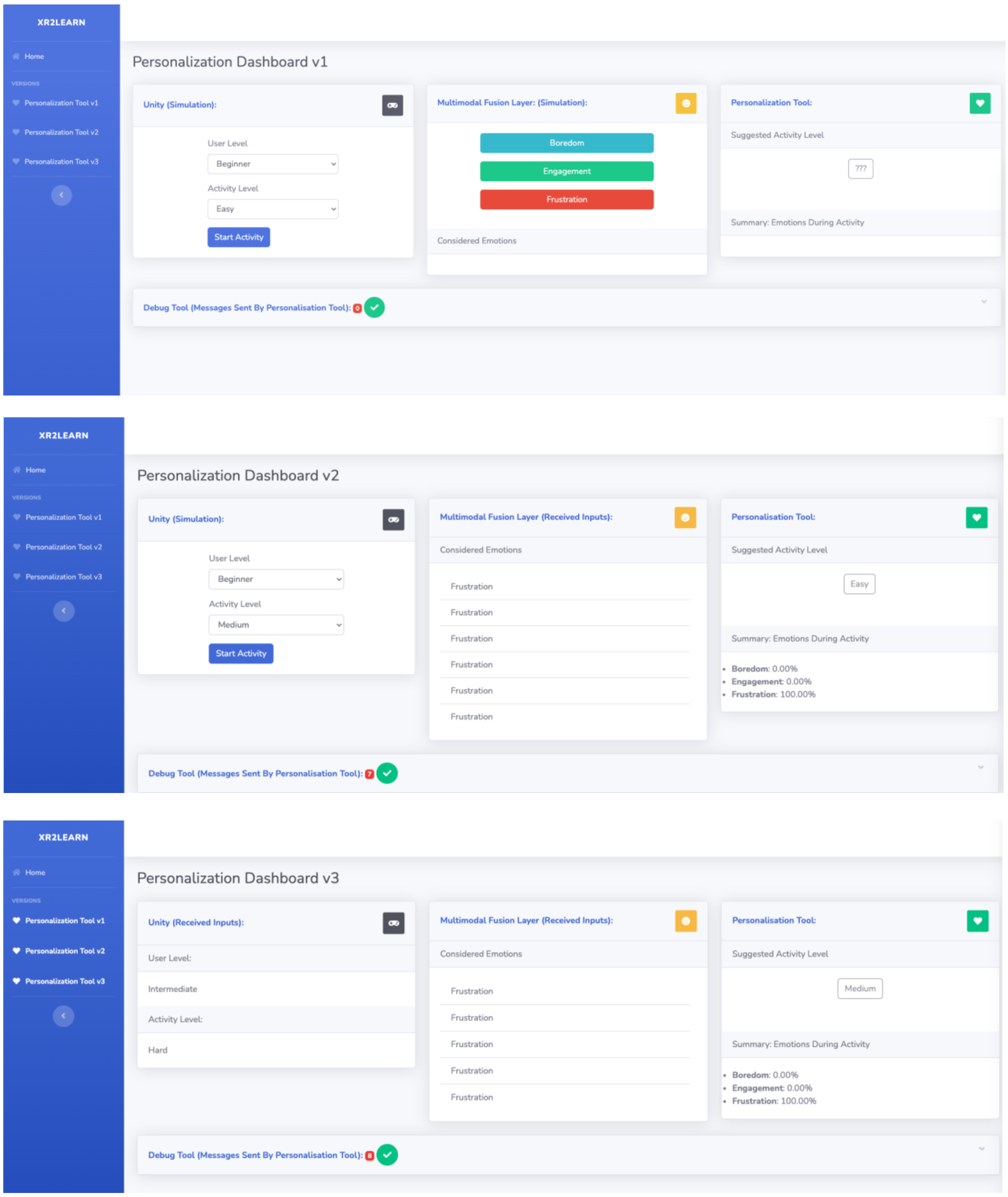
Personalization Dashboard is a web-based graphic interface for users to visualize the communication between the Personalization Enablers (i.e., Inference Tools and Personalization Tool outputs) and a VR application. This enabler, not included in the initial plan, can also simulate parts of the system that have not yet been integrated with the Personalization Tool, making it a valuable debugging resource.

Initially, the Personalization Dashboard had two versions: the first version (v1), the Inference Tools and VR (Unity) application outputs were simulated. This version (v1) helps visualize the Personalization Tool output and test its personalisation heuristic in a streamlined manner without needing additional integration. Utilizing this version facilitates the process of testing and experimenting with the Personalization Tool in a controlled and modularized manner.

In Version 2 (v2), the Personalization Dashboard displays the Personalization Tool and Inference Tools' real outputs and only simulates the VR application outputs. This showcases the integration of Personalization Enablers inner components without needing to integrate with a Unity VR application. Utilizing this version facilitates the process of testing and experimenting with the integration of Personalization Enablers and communication of components, such as Inference Preprocessing, emotion classification, and Multimodal Fusion components.

During M15-M26, we included an additional Personalization Dashboard version. In Version 3 (v3), no component is simulated anymore, and the Personalization Dashboard serves as a graphic display of the messages exchanged by Inference, Personalization Tool, and a VR application. Utilizing this version facilitates testing and visualizing a full integration of Personalization Enablers with VR Unity applications.

Figure 15 displays screenshots of the Personalization Dashboard, including its three versions.


XR2LEARN

Home

VERSIONS

Personalization Tool v1

Personalization Tool v2

Personalization Tool v3

Personalization Dashboard v1

Unity (Simulation):

User Level: Beginner

Activity Level: Easy

Start Activity

Multimodal Fusion Layer: (Simulation):

Boredom

Engagement

Frustration

Considered Emotions

Personalization Tool:

Suggested Activity Level

???

Summary: Emotions During Activity

Debug Tool (Messages Sent By Personalisation Tool): 0

XR2LEARN

Home

VERSIONS

Personalization Tool v1

Personalization Tool v2

Personalization Tool v3

Personalization Dashboard v2

Unity (Simulation):

User Level: Beginner

Activity Level: Medium

Start Activity

Multimodal Fusion Layer (Received Inputs):

Considered Emotions

Frustration

Frustration

Frustration

Frustration

Frustration

Frustration

Personalisation Tool:

Suggested Activity Level

Easy

Summary: Emotions During Activity

- Boredom: 0.00%
- Engagement: 0.00%
- Frustration: 100.00%

Debug Tool (Messages Sent By Personalisation Tool): 7

XR2LEARN

Home

VERSIONS

Personalization Tool v1

Personalization Tool v2

Personalization Tool v3

Personalization Dashboard v3

Unity (Received Inputs):

User Level:

Intermediate

Activity Level:

Hard

Multimodal Fusion Layer (Received Inputs):

Considered Emotions

Frustration

Frustration

Frustration

Frustration

Frustration

Personalisation Tool:

Suggested Activity Level

Medium

Summary: Emotions During Activity

- Boredom: 0.00%
- Engagement: 0.00%
- Frustration: 100.00%

Debug Tool (Messages Sent By Personalisation Tool): 8

Figure 15. Screenshots displaying Personalization Dashboard Version 1 (top), Version 2 (middle) and Version 3 (bottom).

3.5.2. Command Line Interface (CLI)

The Command Line Interface (CLI), described in detail in Deliverable D3.2 XR2learn enablers (M14), Section 3.5, is an additional enabler created to streamline Personalization Enablers' functionalities. Designed with a higher level of abstraction to facilitate the use of Personalization Enablers, CLI includes *simplified commands*, a *single configuration file*, and *pre-setup scripts* and *configurations* for some of the most common use cases.

CLI received updates to support the body tracking modality implementation, near real-time inference tools, and multimodal fusion capability. Utilizing CLI, users can access any Personalization Enabler separately or run end-to-end training, inference and/or personalization pipelines. With the new release, CLI commands have been updated; the prerequisites and installation instructions remain the same, as reported in Deliverable D3.2 version 1 (M14). The following subsection presents the updated basic user manual for the Command Line Interface (CLI).

Basic User Manual

To access the Personalization Enablers' functionalities through CLI, two elements are needed:

1. **CLI commands** and **options**
2. A ***configuration.json* file** (a personalized JSON configuration file path as an option to the CLI command can be provided; if no JSON configuration file path is provided, the default file ./configuration.json is used - available in the relevant repository).

The general command format to use Personalization-CLI is:

```
python xr2learn_enablers_cli/xr2learn_enablers.py [OPTIONS] [COMMAND] [OPTIONS]
```

For help with the options and commands, access a list of arguments and their description with:

```
python xr2learn_enablers_cli/xr2learn_enablers.py --help
```

Command list:

Training (for any supported modality, i.e., audio, bm, body-tracking):

- Audio (RAVDESS data format)

```
python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id
development-model train --dataset ravdess --features_type ssl --
ssl_pre_train encoder_fe --ed_training true
```

- Bio-measurements (Magic XRoom data format)

```
python     xr2learn_enablers_cli/xr2learn_enablers.py     --experiment_id
development-model train --dataset Xroom --modality bm --features_type ssl
--ssl_pre_train encoder_fe --ed_training true
```

- Body Tracking (Magic XRoom data format)34

```
python     xr2learn_enablers_cli/xr2learn_enablers.py     --experiment_id
development-model  train  --dataset  Xroom  --modality  body-tracking  --
features_type none --ssl_pre_train none --ed_training true
```

Inference (Predict Audio modality with files reading/writing, not Pub/sub protocol):

```
python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id
development-model predict --dataset ravdess
```

Inference (near real-time with Pub/Sub Protocol):

Bio-measurement and body tracking uni modality

```
python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id
development-model run-dashboard --modality bm

python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id
development-model run-dashboard --modality body-tracking
```

Bio-measurement and body tracking multimodality (multimodal fusion)

```
python xr2learn_enablers_cli/xr2learn_enablers.py --experiment_id
development-model run-dashboard --modality bm --modality body-tracking
```

Go to the URL http://127.0.0.1:8080 to access the Personalization Dashboard

The commands above for inference near real-time will run the Personalization Dashboard by default, and the result of the near real-time prediction for unimodal or multimodal prediction can be visualized in the Dashboard.

Note: To run Dashboard for one modality or multiple modalities, the command and the configuration must match, i.e., if the dashboard command is run with one modality, the `` `configuration.json` `` file must reflect the same modality.

Stop Personalization Dashboard (and all the Personalization Enablers services running)

```
python xr2learn_enablers_cli/xr2learn_enablers.py stop-dashboard
```

Available versions

Table 6 lists the versions released during M15-M26, and the full list of corresponding changes for each version can be accessed in the Command Line Interface changelog via the following link:

https://github.com/XR2Learn/Personalization-Enablers-CLI/blob/master/CHANGELOG.md

Table 6. Command Line Interface released versions and dates.

Version	**Release Date**
v0.5.0	**2024-03-12**
v0.5.1	**2024-04-10**
v0.6.0	**2024-07-25**
v1.0.0	**2024-10-28**

All the released versions can be accessed and downloaded at: https://github.com/XR2Learn/Personalization-Enablers-CLI/releases

3.5.3. Personalization Enablers Template

Description

Personalization Enablers consist of multiple components working together. This intricate architecture can present challenges for developers when they are tasked with creating new enablers. To facilitate the implementation of new enablers by the community, an additional enabler that serves as a template has been developed. This template makes the creation of new enabler components much easier, facilitating the development process and boosting the adoption of the XR2Learn solution.

The Personalization Enablers Template is available as an open-source project on GitHub. Developers can use a single command to create a placeholder skeleton for a new enabler, already pre-configured to communicate with the other Personalization Enablers. This setup includes files and folder structure, Docker configuration, unit tests, versioning, and documentation.

By utilizing the Personalization Components Template, developers can generate the entire structure of a new enabler with just one command. They will only need to modify the code file to implement the desired functionality for the new enabler, significantly reducing the effort required on the software architecture side.

Prerequisite and Installation

This component uses the **cookiecutter Python library**[11]. To install this library, run the following command:

```
pip install --user cookiecutter
```

There is no further installation required.

Basic User Manual

1. Run the command
   a. `cookiecutter git@github.com:XR2Learn/Personalization-Components-Template.git`

2. Answer the prompt questions:
   a. `"project_name": Name of the Component, following the format: "SSL Features Extraction BM Modality",`
   b. `"description": Description of the component,`
   c. `"project_slug": e.g., "ssl_features_extraction_bm_modality",`
   d. `"component_folder": e.g., "SSL_Features_Extraction_BM_Modality",`
   e. `"parent_component": Parent Component Name, e.g., "Pre_precessing",`
   f. `"main_python_file": Python entry-point script name without .py format, e.g., generate_features.py`
   g. `"service_name": e.g., "ssl-features-generation-bm",`
   h. `"current_version": Repository current version in the format: "0.1.0"`

3.6. RESEARCH PUBLICATIONS

While the research phase preceding the development of Emotion Recognition enablers was finalized in M14, as described in Deliverable 3.2 XR2Learn enablers, Section 3.1, the outcomes of the conducted research led to the following scientific publication:

- Khaertdinov, B., Jeruis, P., Sousa, A., Hortal, E. (2024) Exploring Self-Supervised Multi-view Contrastive Learning for Speech Emotion Recognition with Limited Annotations. Proc. Interspeech 2024, 4708-4712, doi: 10.21437/Interspeech.2024-860

[11] https://cookiecutter.readthedocs.io/en/latest/installation.html

4. DATA ACQUISITION

4.1. MAGIC XROOM: DATA COLLECTION TOOL

4.1.1. Magic XRoom Summary up to M14

Magic XRoom is a virtual reality application developed to collect multimodal sensor data while eliciting specific emotional responses. Figure 16 shows an overview of the Magic XRoom virtual room. The application allows users to play games that are designed to elicit emotions based on the Theory of Flow, namely engagement, frustration, or boredom. Magic XRoom also asks users to provide feedback describing their affective response according to the mentioned categories. This feature allows further processing of collected data annotated based on the users' responses. Magic XRoom contains the four following games with varying difficulty levels:

- **Stacking Cubes.** The user is asked to stack four cubes within a time limit while facing certain challenges. Failed levels must be repeated until successful (Figure 17).
- **Color Words.** The user selects a cube based on the color name shown on the screen, ignoring the font's color. The game gets harder as more cubes appear and the time for actions decreases. The game ends upon selecting the wrong cube or running out of time.
- **Canvas Painting.** The user must draw specific shapes within a defined area while avoiding mistakes in a limited amount of time. A limited number of errors are allowed.
- **Tower of Hanoi.** The user matches a target configuration of disks using an interactive set of rods and disks. The rods have different size capacities (3, 2, and 1 disks, from left to right). Moves are limited, and the task must be completed within a time constraint (Figure 18).

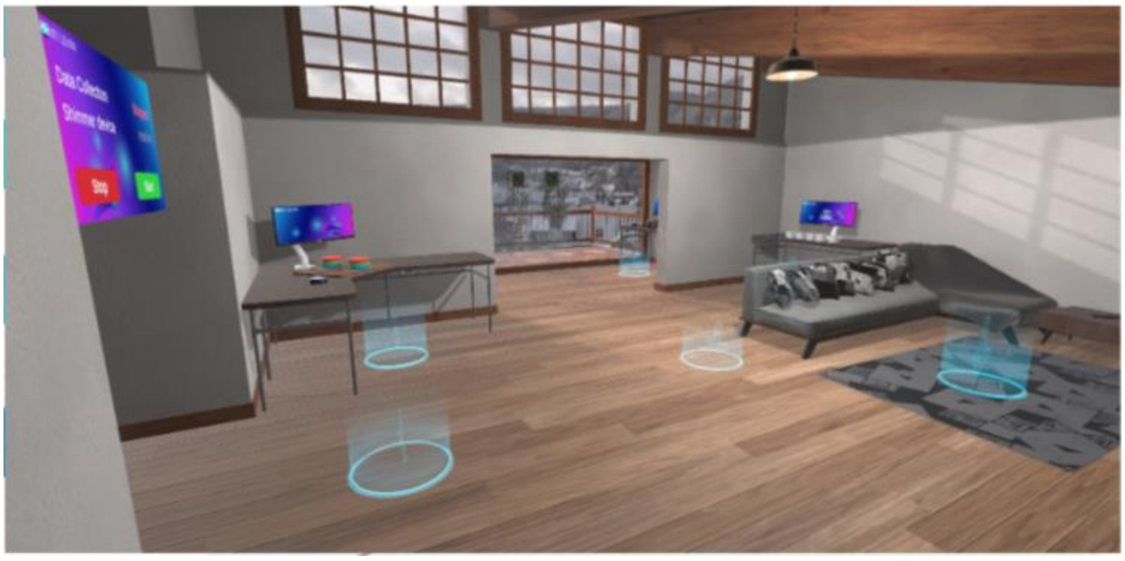


Figure 16. Screenshot displaying Magic XRoom's virtual room with the four interactive scenarios.

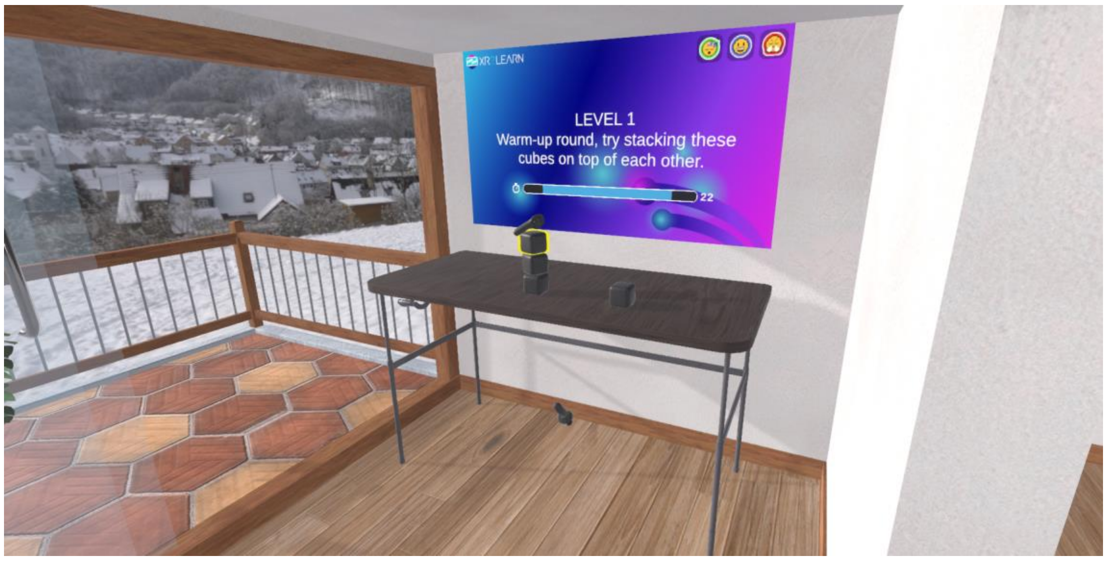

XR2LEARN
LEVEL 1
Warm-up round, try stacking these cubes on top of each other.
22

Figure 17. View of the stacking cubes scenario

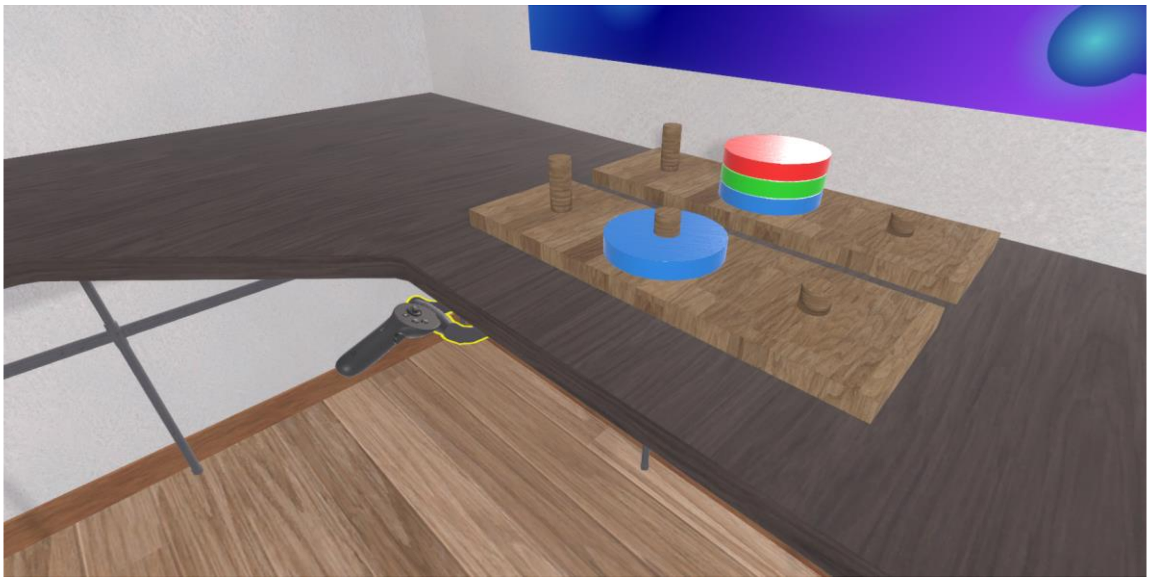


Figure 18. Detailed view of the Tower of Hanoi scenario and the interactive handle used to adjust the desk height

Magic XRoom collects VR headsets' and VR controllers' data, capturing positional and rotational information. It also supports integration with external sensors, namely Shimmer 3 GSR+, and devices for eye and mouth tracking. Therefore, it is possible to acquire physiological and behavioral signals, such as galvanic skin response (GSR), heart rate, pulse wave, and eye/lip movement features. All data is synchronized and exported in CSV format, with adjustable parameters like sampling frequency and buffer sizes that can be provided in the configuration file. A detailed summary of the configuration file structure is available in the first version of the deliverable and the GitHub repository of the tool: https://github.com/XR2Learn/magic-xroom

4.1.2. Magic XRoom Updates M15-M26

The Magic XRoom was deployed and tested during the initial data collection phase (M10-M13). This process identified a series of issues and gathered valuable user feedback, which guided the subsequent implementation of fixes and enhancements.

The primary challenge following the first data collection was to enhance the application's immersiveness. To address this, several assets and props were added to create an environment that struck a balance between being engaging and welcoming, ensuring it supported users without disrupting their tasks.

However, adding these assets and props introduced an additional challenge related to maintaining the environment's quality. It became evident that several changes were necessary, including adjustments to lighting, reflections, materials, textures, surroundings, additional props, and shadows. These modifications not only improved the visual fidelity but also required updates to user interactions within the VR environment to ensure a seamless experience.

Moreover, the fact that the Magic XRoom is a VR application added another layer of complexity. The introduction and placement of these elements had to be carefully planned to accommodate users of varying heights and levels of experience, ensuring accessibility and usability for a diverse audience.

- Version 1.1 of the Magic XRoom included a revamped environment and changes to the overall feel of the scenarios. Textures and lights were adjusted to improve realism, and several quality-of-life changes were added to the user interactions.
- Version 1.2 experienced a restructuring of the user interface, including a shared and improved theme, updated animations, improvement to the feedback mechanics (Figure 19), and, last but not least, a better introduction to each scenario.
- Version 1.3 was released just before the beginning of the second data collection (M23 - M24) and included bug fixes and adjustments to the output data collected following internal tests and feedback from UM.

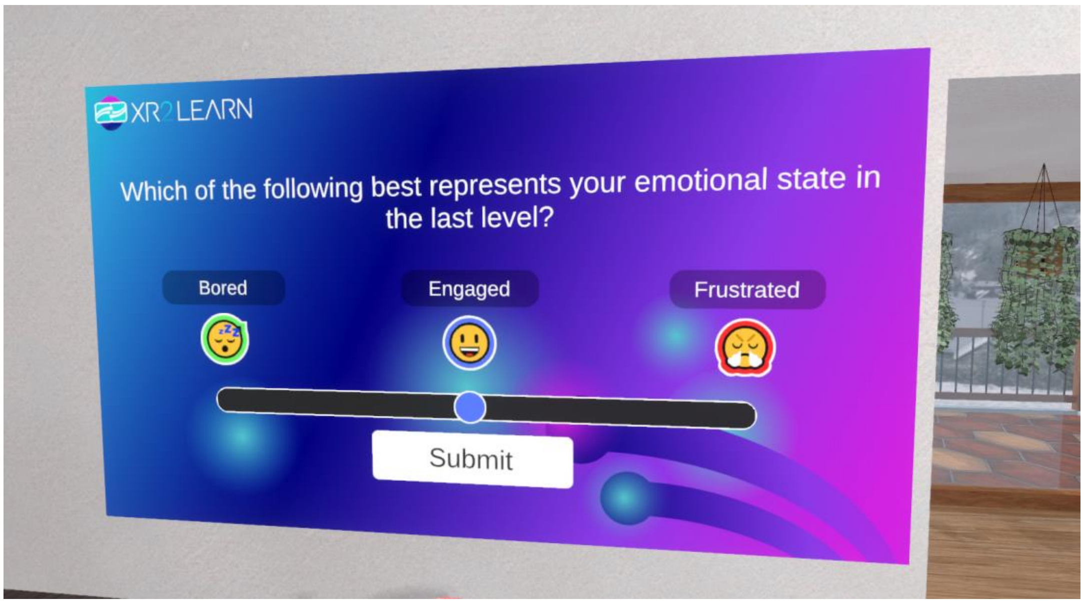

XR2LEARN
Which of the following best represents your emotional state in the last level?
Bored
Engaged
Frustrated
Submit

Figure 19. Detail of the updated user interface showing the feedback panel, which includes a (continuous) slider instead of 3 (discrete) buttons, easing the selection of the user emotion.

In addition to the changes mentioned above, a refactoring of the overall codebase was undertaken to improve the quality of the project hosted on our GitHub repository. A clear separation between support libraries/frameworks and the data collection tool code provides an easier approach to understanding the application's inner workings. This process is part of a continuous effort to maintain a high standard of code quality and will extend to future updates as the project evolves.

Table 7 lists the versions released during M15-M26.

Table 7. Magic XRoom released versions and dates.

Version	**Release Date**
v1.1.0	**2024-06-27**
v1.2.0	**2024-09-27**
v1.3.0	**2024-11-07**

4.2. DATA COLLECTION

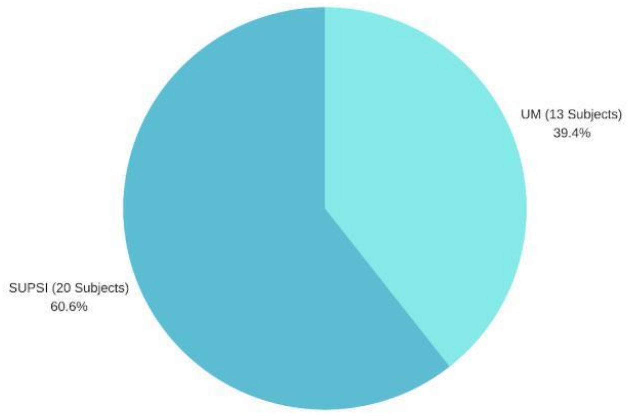
4.2.1. Data Collection Summary up to M14

The first pilot sessions were conducted in M10-13 at UM and SUPSI premises and aimed to test Magic XRoom and the proposed data collection protocol with 31 participants. Furthermore, the collected dataset was used to adjust the personalization enablers pipeline to the Magic XRoom data format. Table 8 summarizes the first data collection pilots, and Table 9 shows the gender distribution of the first pilot participants.

A preliminary analysis of the pilot session data revealed highly unbalanced data, as shown in Figure 20, which motivated some changes in the data collection protocol in order to elicit the least present classes, i.e., *boredom* and *anxiety*.

Table 8. Data collection pilots' statistics.

Dates	**Location**	**Number of participants**	**Purpose**
January-February 2024	SUPSI	20	Testing Magic XRoom
January 2024	UM	13	Testing Magic XRoom, obtain data input format to implement enabler components for bio-measurement modality.



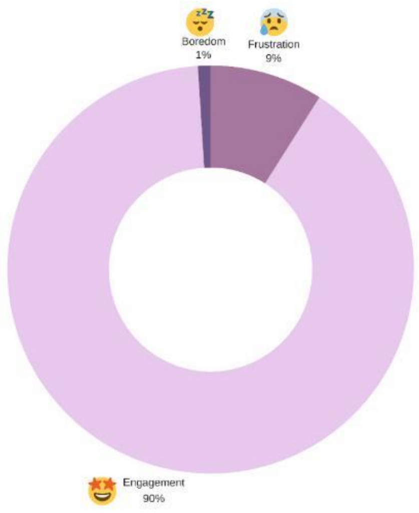


Figure 20. Preliminary analysis of the pilot session data showing the distribution of engagement levels

Table 9. Gender distribution for the first data collection pilots.

Location	Participants	Males	Females
SUPSI	20	17 (85%)	3 (15%)
UM	13	10 (76.92%)	3 (23.08%)

The data collection study (including consent forms and questionnaires) has been approved by SUPSI's and UM's ethics committees before commencing the data collection pilot sessions. Additionally, we adhered to the data management guidelines outlined in D1.2 “Data Management Plan”.

4.2.2 Data Collection Updates M15-M26

Data collected in the project's first year primarily served as a template for the Personalization Enablers. It informed the format and structure of the data needed to develop its components. During months 15-26, UM and SUPSI collected the second pilot data to gather data for training models within the personalization tools.

4.2.2.1. Data Collection Protocol

For the second data collection pilot, SUPSI and UM recruited 45 participants.

The data collection protocol from the first pilot reported in the first version of D3.2 has been extended to address the known challenges. Specifically, the new data collection contains three parts: baseline measurements, calibration rounds, and main data collection.

The baseline measurements part involves the participant sitting in silence for two minutes while wearing all VR equipment and sensors. This step ensures that data is being recorded successfully and gives participants an opportunity to remain calm and not feel agitated during the study.

Next is the calibration round, where participants play each of the four games for one minute each. This allows them to understand the mechanics and physics of each game, ask questions about the games, and become accustomed to the VR environment.

Finally, the main data collection occurs, with participants playing each game for approximately two and a half minutes. By this stage, they have learned how to play the games and are more familiar with the extended reality (XR) experience.

To participate in data collection, subjects were required to read a participant information sheet, sign the consent forms, and complete a pre-study questionnaire. Furthermore, after the study, the participants were encouraged to complete an anonymous post-study survey, where they could share their experiences.

The following shared folder contains all resources utilized during data collection, including the participant information sheet, the consent form template, the complete data collection protocol, and pre- and post-study questionnaires links, and the poster used for participants recruitment:

https://surfdrive.surf.nl/files/index.php/s/sT2gpg31nZ8Xmmg

4.2.2.2. Collected Data Analysis

As shown in Table 10, 45 participants, including 34 males and 11 females, participated in the data collection experiments at SUPSI and UM within two months. This section presents a brief overview of the collected dataset.

Table 10. Data collection overview.

Dates	Location	Participants	Amount of data	Amount of labeled data	Purpose
November-December 2024	SUPSI	23	7.5 hours	5.5 hours	Collect data for training AI models.
November-December 2024	UM	22	10.5 hours	4.5 hours	

Survey analysis

During the second data collection, we focused our recruitment efforts on attracting female participants to achieve a more balanced gender distribution in our dataset. However, achieving this balance has been challenging, as most of the participants come from computer and engineering departments, which still have a significant gender imbalance. These fields tend to have a higher number of male students at both UM and SUPSI. Despite these challenges, we successfully increased the percentage of females in our dataset for the second data collection from approximately 23% to nearly 37% at UM. Table 11 shows the age and gender distribution of the second pilot participants.

Additionally, we conducted exploratory data analysis and reported multiple statistics for participants who completed the pre- and post-study questionnaires in this section.

Table 11. Gender and age statistics.

Location	Mean age (std)	Males	Male mean age (std)	Females	Female mean age (std)
SUPSI	26.8 ± 8.3	20 (86.95%)	27.1 ± 8.4	3 (13.05%)	25.3 ± 9.2
UM	27.2 ± 5.8	14 (63.64%)	27.8 ± 4.8	8 (36.36%)	26.5 ± 7.0

Figure 21 shows the self-reported proficiency of subjects with computers and VR technology. While most participants identified as being proficient with computers and electronic devices, they are generally not very experienced with VR technology.

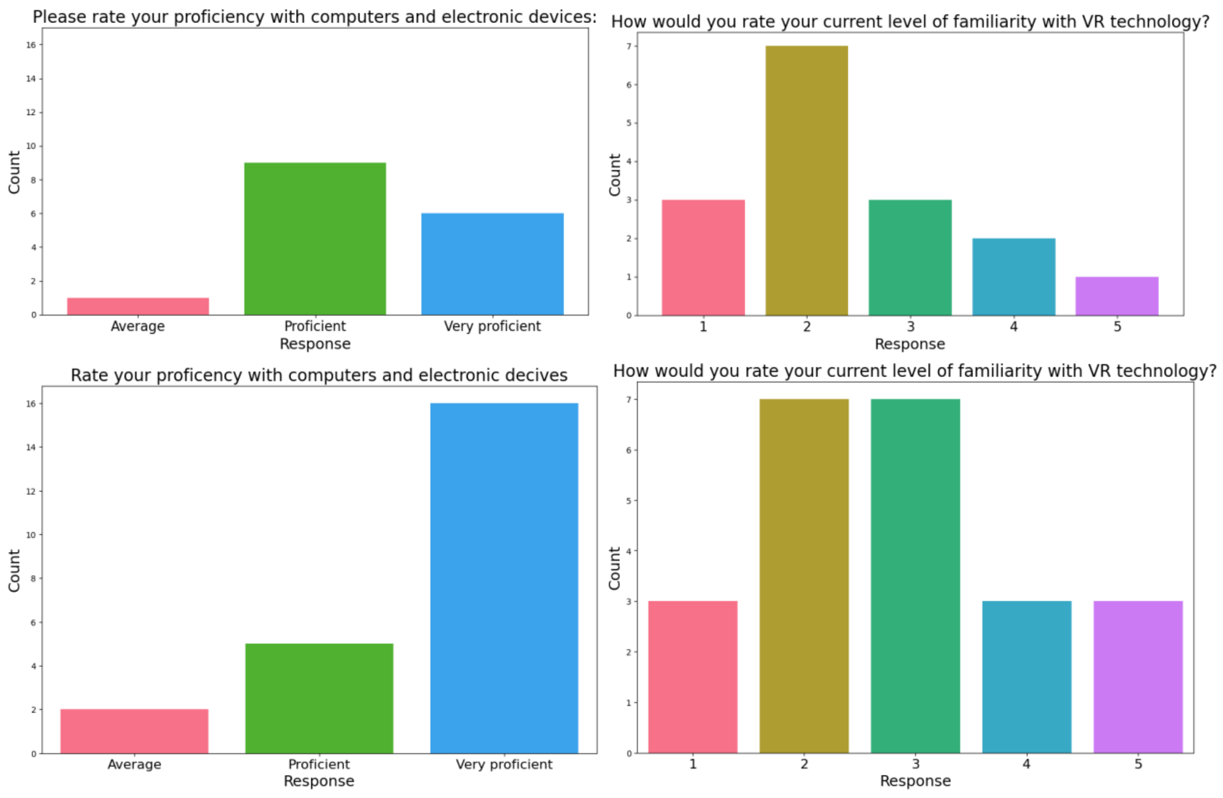


Figure 21. Proficiency of users with electronic devices and familiarity with Virtual Reality (upper – UM, lower – SUPSI).

According to the anonymous post-study questionnaire, more than half of the participants felt very comfortable during data collection and found this experience satisfying, as shown in Figure 22.

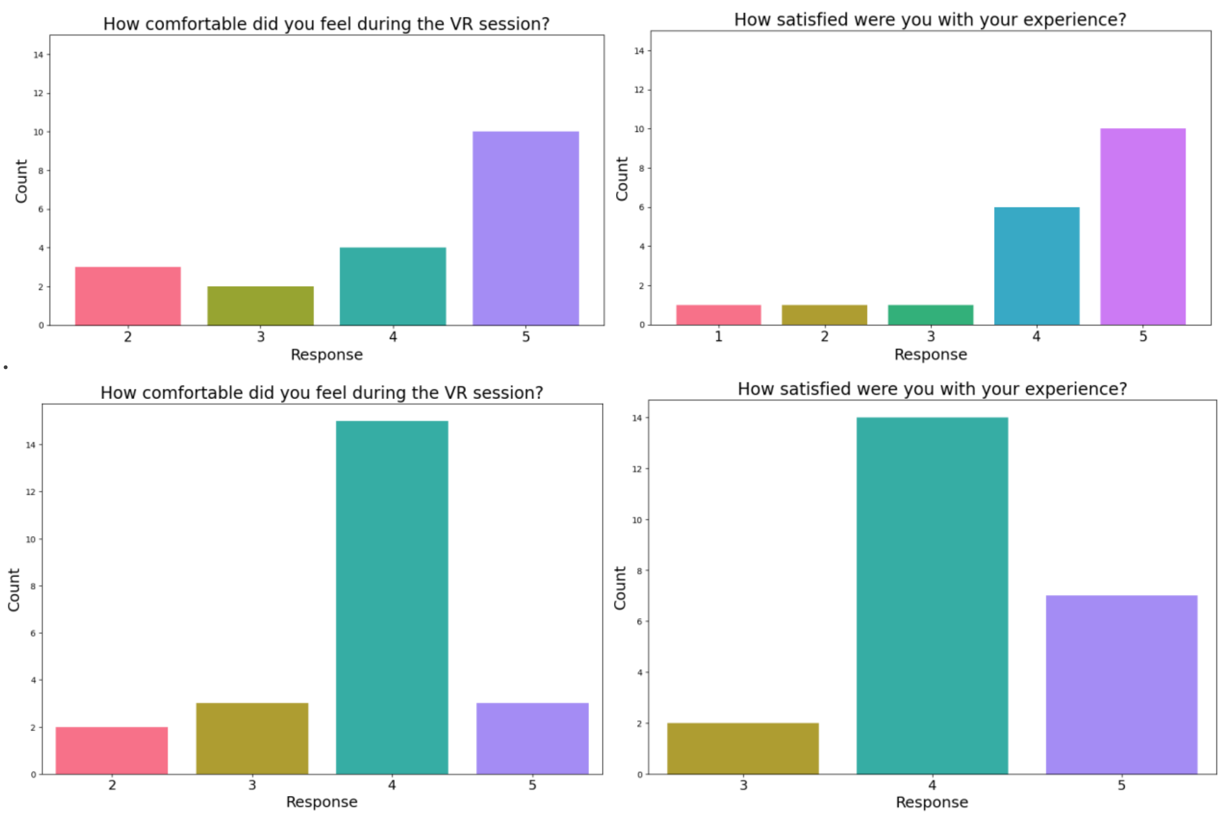


Figure 22. Comfortability and satisfaction levels during data collection (upper – UM, lower – SUPSI).

Nevertheless, a significant fraction of subjects reported no signs of physical discomfort during or after the session (Figure 23). According to their responses, they experienced the following conditions: “Little Nauseous near the end”, “A bit of eye strain at the end of the session”, “Headache in the last minute or two”, “Headset was not very comfortable and when it moved it made me dizzy”, “some numbness where the headset was sat”, “eyes hurting”, “headache, motion sickness, little nausea, if I was in the VR for much longer I would be very uncomfortable”, “dizziness with adjusting the VR clarity”, “I was standing for too long, I started to feel my feet”.

Furthermore, in Figure 23, it can be observed that females reported experiencing more physical discomfort during the study sessions compared to males. This finding contradicts a study by Larson et al. (1999)[12], which found no evidence of gender differences in VR environments regarding females' higher susceptibility to cybersickness. However, it's important to note that the difference in reports of physical discomfort between females is small due to the limited number of female participants. For example, at UM, 5 females reported discomfort while 4 did not; at SUPSI, 2 females reported discomfort and 1 did not. Therefore, although our results contradict those of Larson et al., the difference is not significant enough to draw definitive conclusions. Further studies should focus on gender-based discomfort in VR to provide more insights.

[12] Larson, Peter, et al. "Gender issues in the use of virtual environments." CyberPsychology & Behavior 2.2 (1999): 113-123.

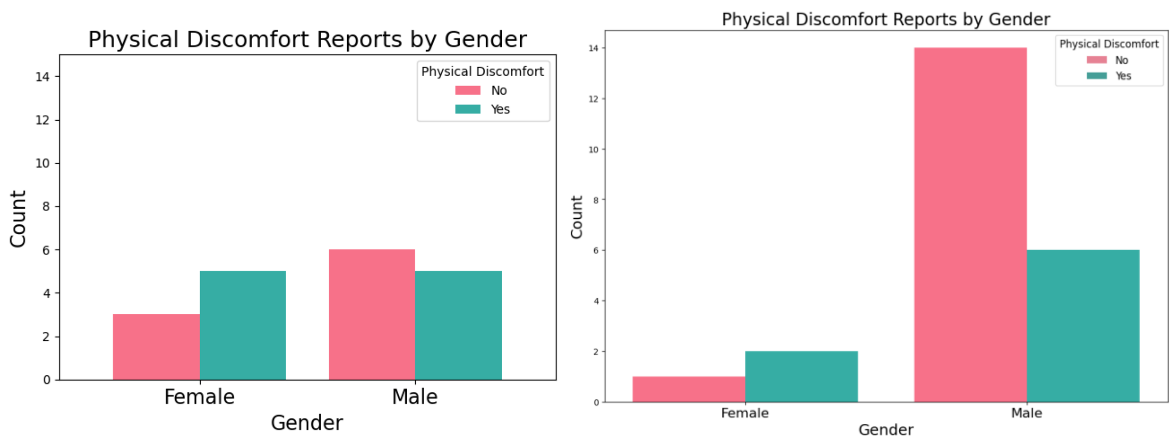


Figure 23. Physical discomfort experienced during or after data collection(left – UM, right – SUPSI).

Label distribution

One of the main issues faced during the first data collection pilot was the highly unbalanced distribution of labels. For the second round of pilots, we have updated the scenarios in Magic XRoom and the data collection protocol to elicit more examples of boredom and frustration. Furthermore, participants report their feedback on a continuous scale—this can be used to apply different thresholding functions to assign categorical levels. This section analyses the label distribution obtained in the dataset collected within the second pilot.

During the first data collection pilots before M14, it was noticed that users often (around 90% of feedback messages) report engagement, which results in a highly skewed distribution of labels for training (as shown in Figure 20). In the newer versions of Magic XRoom, users provide feedback on a continuous scale between 0 and 1, where zero corresponds to boredom, 0.5—engagement, and 1—frustration (Figure 19). We present the distributions of the reported values per subject in Figure 24.

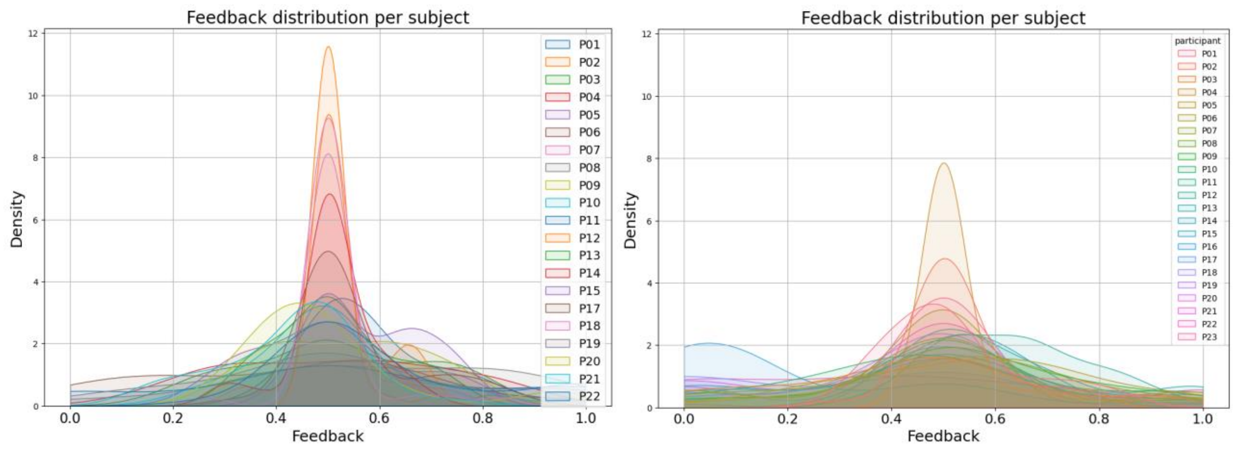


Figure 24. Distribution of continuous engagement levels per subject (left – UM, right – SUPSI).

As can be observed in Figure 24, the values for multiple subjects are still centered around 0.5 (engagement). Figure 25 illustrates the distribution of continuous engagement levels aggregated for all subjects.

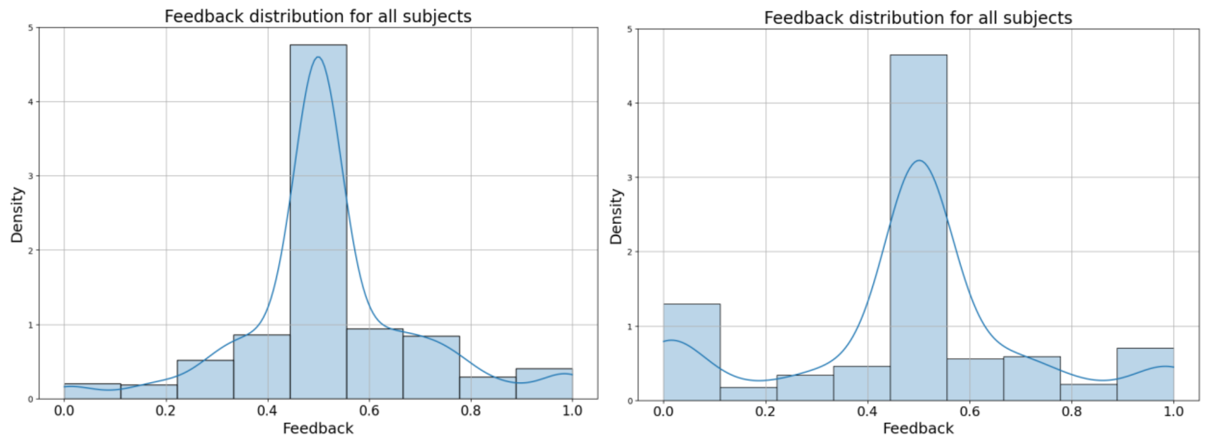


Figure 25. Distribution of continuous engagement levels for all subjects (left – UM, right – SUPSI).

Furthermore, in the current version of training and inference tools, continuous engagement levels can be converted into distinct categories based on provided boundaries. Figure 26 illustrates the distribution of categories with equally wide boundaries per category (Bored [0.0, 0.33], Engaged [0.34, 0.66], Frustrated [0.67, 1]) and narrower engagement borders (Bored [0.0, 0.44], Engaged [0.45, 0.55], Frustrated [0.56, 1]). The flexible thresholding has also been integrated into Training Tools and can be used to train models to distinguish high-engagement episodes.

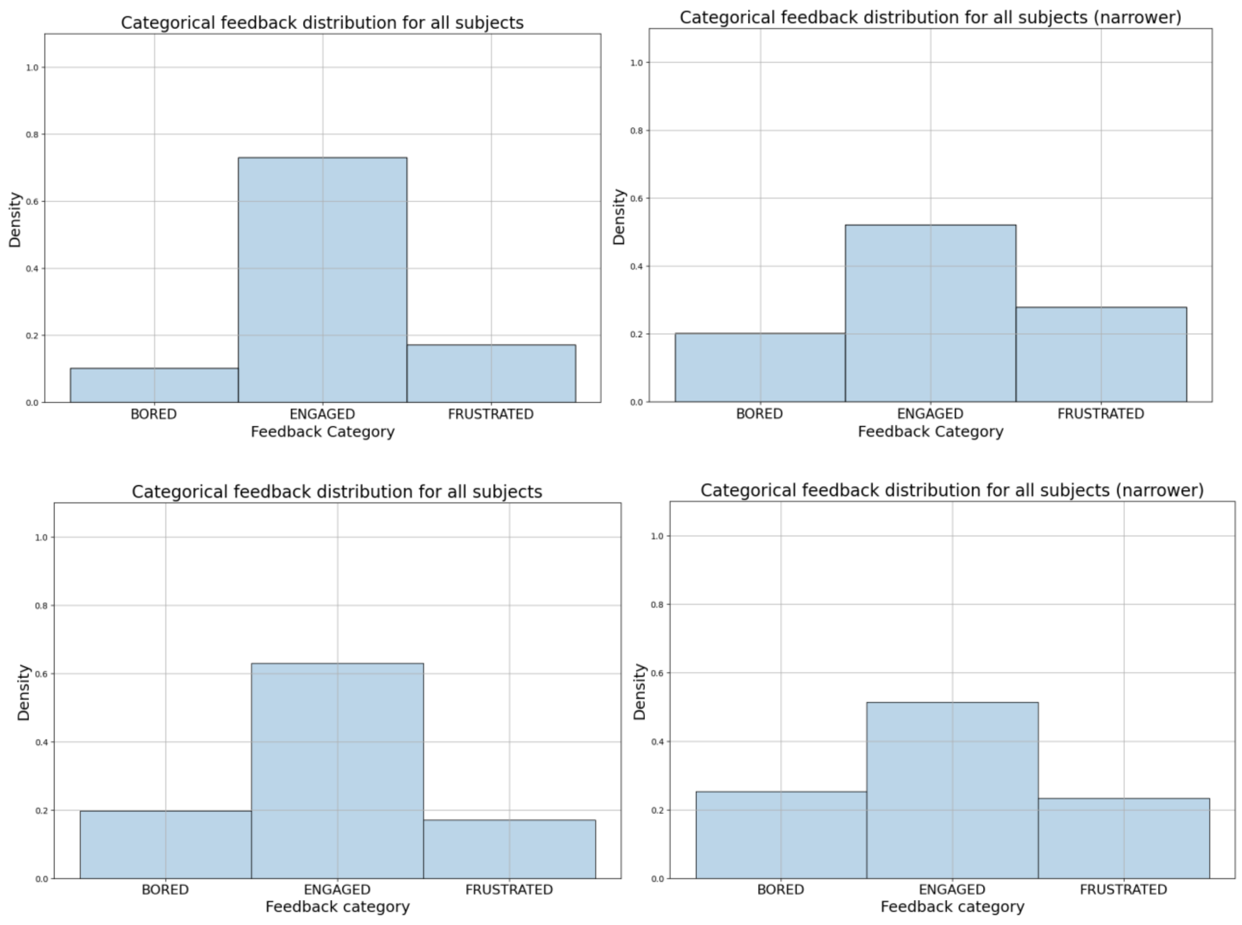


Figure 26. Distribution of categorical engagement levels for all subjects (upper – UM, lower – SUPSI).

We computed the distribution of feedback categories per game and visualized it in Figure 27. Interestingly, despite different datasets collected at different locations (UM and SUPSI), similar trends in labels can be noticed. For example, it is clear that users find the Color Words game more engaging, while Canvas Painter and Tower of Hanoi make users feel frustrated and bored more often than other games, respectively. This analysis can further inform specific emotion elicitation while collecting data with Magic XRoom in the future.

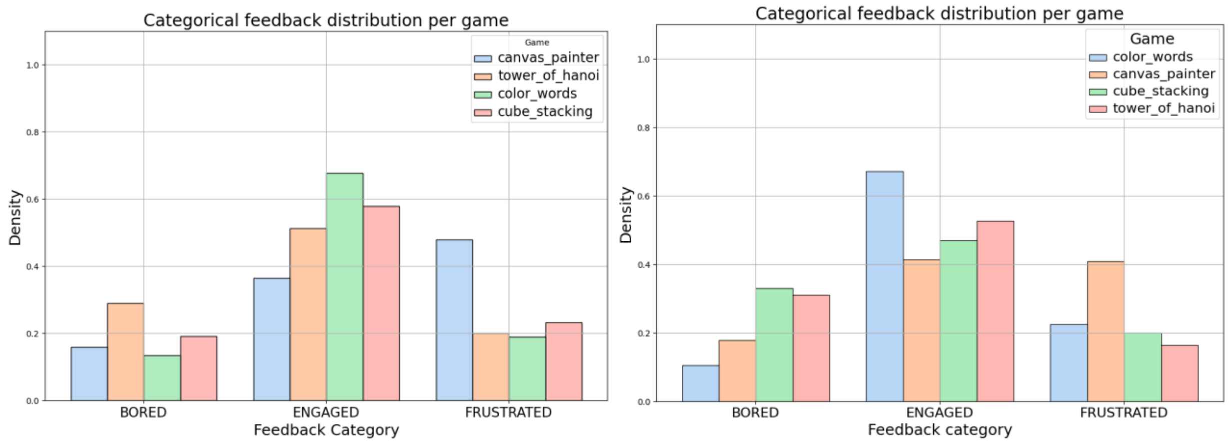


Figure 27. Distribution of categorical engagement levels for each scenario in Magic XRoom (left – UM, right– SUPSI).

Overall, subjects still mainly reported engagement. Nevertheless, adjusting the boundaries for each category can balance the distribution of labels and isolate segments with the highest engagement levels. In future work, defining individual boundaries and/or normalizing engagement levels per subject might also be an interesting approach to explore.

4.2.2.3. Training Models with Collected Data

The dataset collected with Magic XRoom is used to train and evaluate models from the Training Tools (Section 3.2). To evaluate how models generalize to new subjects, we split data into three sets based on subjects. The test set contains data from 10% of subjects and is used to report the final performance of models. The remaining data is used for training and validation purposes: 10% of the remaining subjects are used for validation between epochs, and the rest are used for training.

During the initial experimentation, we noticed that the models suffer from class imbalance and are biased toward the majority class, resulting in poor performance. In this case, regardless of input, the model almost always predicts the majority engagement category. We denote it as a *naive model*.

To address this issue, we conducted multiple experiments in which we tried to tune hyperparameters. Namely, we experimented with different overlapping proportions in data segmentation, boundaries to transform continuous engagement levels into categorical ones, class weights for loss function, and other hyperparameters related to model architectures. The best results obtained in this series of experiments are shown in Table 12.

Table 12. Results of model evaluation. All metrics were computed on the test subjects, which data were unseen during training.

Dataset	**Modality**	**Model**	**F1-score (macro)**[13]	**Recall (macro)**	**Precision (macro)**
SUPSI	Bio-measurements	Naive	0.20	0.33	0.15
		Best	0.40	0.41	0.41
	Body-tracking	Naive	0.19	0.25	0.15
		Best	0.46	0.52	0.46
UM	Bio-measurements	Naive	0.26	0.33	0.23
		Best	0.36	0.39	0.37
	Body-tracking	Naive	0.23	0.33	0.17
		Best	0.35	0.39	0.33

According to the obtained result, the models in Training Tools outperform the naive baseline. Nevertheless, the performance of models is still sub-optimal. There are a couple of reasons that might cause such behavior. First, it is a challenging task for a model to generalize well to unseen subjects in validation and test data with emotion annotations that are subjective. Exploring a more personalized approach that calibrates and re-train models for each subject might be an interesting research line. Second, the bio-measurement sensor is attached to the fingers of the subject, which may cause motion artifacts (noise) in raw data. While it is possible to measure galvanic skin response and blood volume pulse at other locations on the human body, most sensor manufacturers do not give access to raw data and/or do not provide API for real-time streaming. The investigation of the impact of the placement of the sensors is planned as future work.

[13] Macro averaging: compute the metric value per class and average across classes.

5. INTEGRATION ENABLERS AND BEACON APPLICATIONS

One important objective for Task 3.2, sub-phase two, is demonstrating the integration between enablers and beacon applications. From a technical perspective, integrating Personalization Enablers and a beacon application requires developing a VR application, *i.e., a beacon application,* that can **(1)** collect and write data from users (number 1 in **Figure 28),** and **(2)** send and receive information to and from the Personalization Enablers (number 2 in **Figure 28)**. The VR application must transmit *contextual information* and receive *suggestions for personalization* from the Personalization Enablers. The user data can then be read and processed by Inference and Personalization Tools to calculate and provide personalized suggestions to the VR application, which in turn will adapt the learning experience based on the suggestions received by the Personalization Enablers.

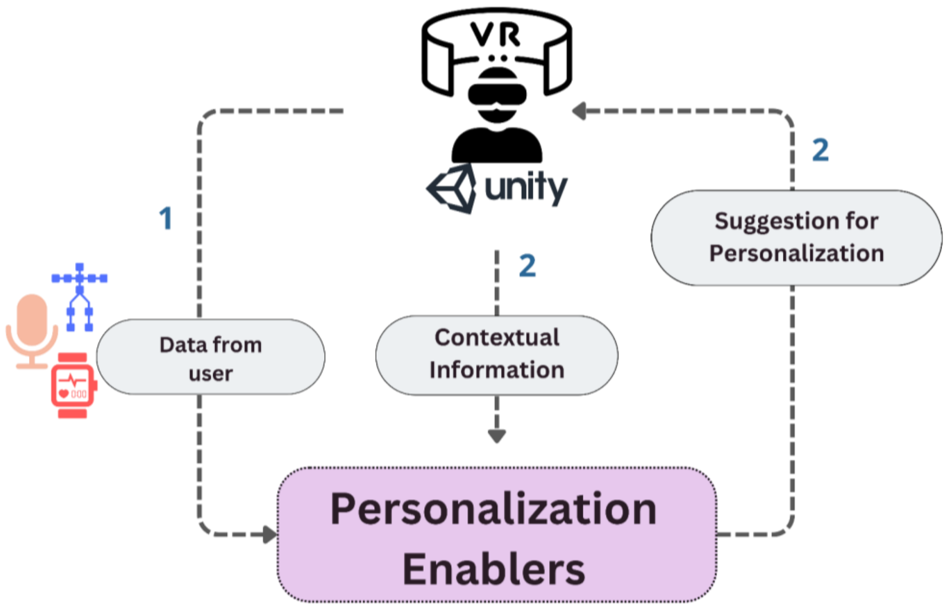


Figure 28. A high-level overview of the integration between enablers and beacon applications. Number 1 represents the functionality of collecting and writing user data, while number 2 represents the functionality of sending and receiving messages from and to the Personalization Enablers.

Therefore, to implement the integration of enablers and beacon applications, we divided the effort into three tasks, as listed below. The first two tasks are under the scope of *Task 3.2*, as they involve enablers preparation, whereas the third task is under the *scope of Task 3.1*, as it involves beacon applications modifications.

1. Develop a Unity Personalization Integration Template (UM);
2. Develop a data collection module/plug-in for Unity applications (SUPSI);
3. Expand Beacon Application 1 - “Laser Cutting Machine” (LS) by:
    a. including *data collection* functionality
    b. including *communication with Personalization Enablers* functionalities, and
    c. automatically adapt the educational content according to the suggestions received from the Personalization Enablers.

In the following sections, we describe what each of the above-listed tasks entails, their role in completing integration between Personalization Enablers and Beacon Applications, and the progress achieved.

5.1. PERSONALIZATION INTEGRATION TEMPLATE

5.1.1. Description

The personalization integration template has been delivered as an open-source GitHub repository containing a Unity example project showcasing how to communicate a Unity Application with the Personalization Enablers. Personalization Enablers and Unity applications communicate by exchanging messages using the Pub/Sub protocol, which was implemented using the Redis technology as a message broker.

Thus, this example includes the source code to:

- Connect to Redis in Unity;
- Create a publisher and a subscriber in Unity;
- Sending and receiving message formats to communicate with the Personalization Enablers;
- A graphic interface, as depicted in Figure 29;
- Documentation on how to set and use this template;
- An already compiled ready-to-use build for Linux x86_64 architecture[14].

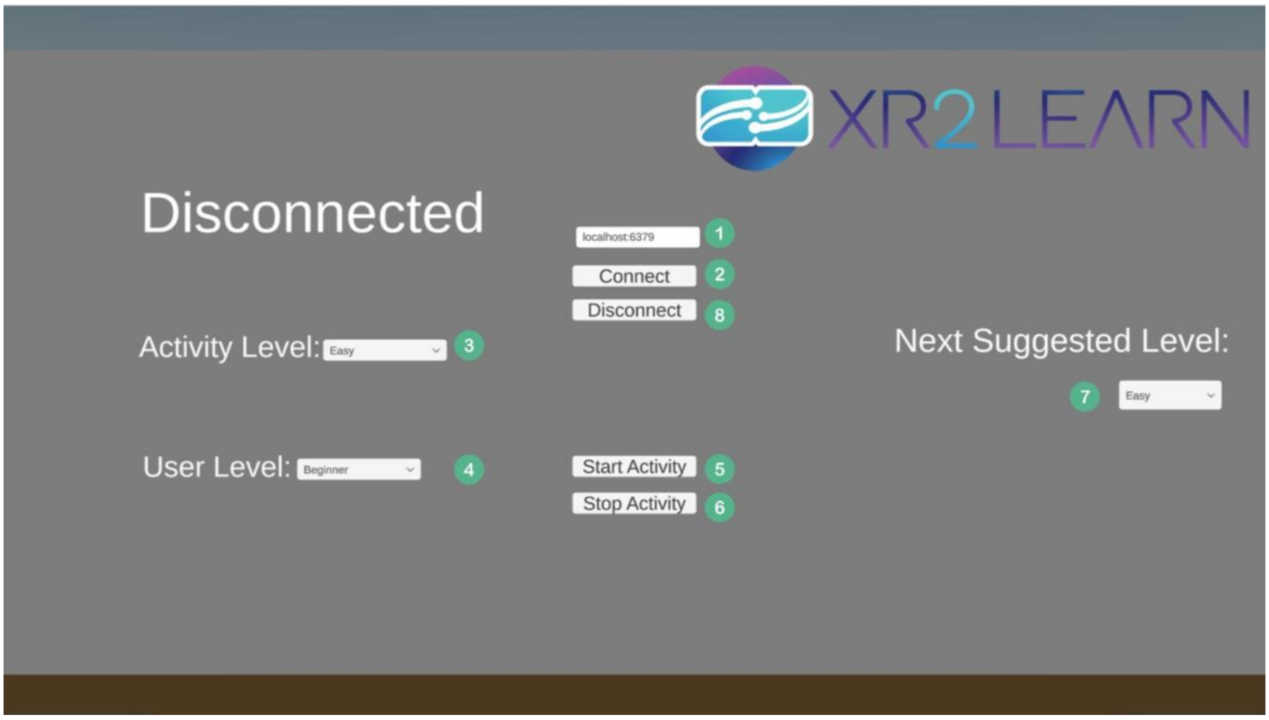


Figure 29. The graphical interface of the Personalization Integration Template (green numbers are not part of the software's interface; Section 5.1.4 describes their functionalities)

This tool facilitates the integration of the Personalization Enablers into third-party Unity applications. To that end, a developer who wants to integrate their VR Unity

[14] https://drive.google.com/drive/folders/1y3j8F7yACtt1lwrk7ARYgFjxEEVgG4f6

application with the Personalization Enablers functionalities should refer to this project and include the source code into their VR Unity application.

5.1.2. Prerequisites

1. NuGets in Unity (to Install NRedisStack);
2. NRedisStack NuGets (to connect Unity to Redis);

5.1.3. Installation

Installing NuGets in Unity (official documentation[15])

Overview install instructions:

In the project, go to *Window->Package Manager->[+]->Add Package From git URL* and paste the URL for the NuGetForUnity:
https://github.com/GlitchEnzo/NuGetForUnity.git?path=/src/NuGetForUnity

Install NRedisStack NuGets

Go to *NuGet For Unity->Online (tab)* and search for *NRedisStack*. Select this package and click on *Install All Selected*.

5.1.4. Basic User Manual

This section contains instructions on how to run the Personalization Integration Template, including the configuration required and basic functionality to test the template as a Unity application.

Running the App

1. Set the Redis connection string to the IP:PORT where the Redis instance is running (Number 1 in Figure 29).
2. Then click on Connect (Number 2 in Figure 29).
3. Select User Level and Activity Level (Numbers 3 and 4 in Figure 29).
4. Click on Start Activity to publish the message that will start an activity (Number 5 in Figure 29).
5. Click on Stop activity to publish the message that will stop the current activity (Number 6 in Figure 29).
6. You will be able to see the Next suggested activity level on the Dropdown to the right (Number 7 in Figure 29).
7. Clicking on Disconnect will end the connection to the Redis instance (Number 8 in Figure 29).

Redis Connection

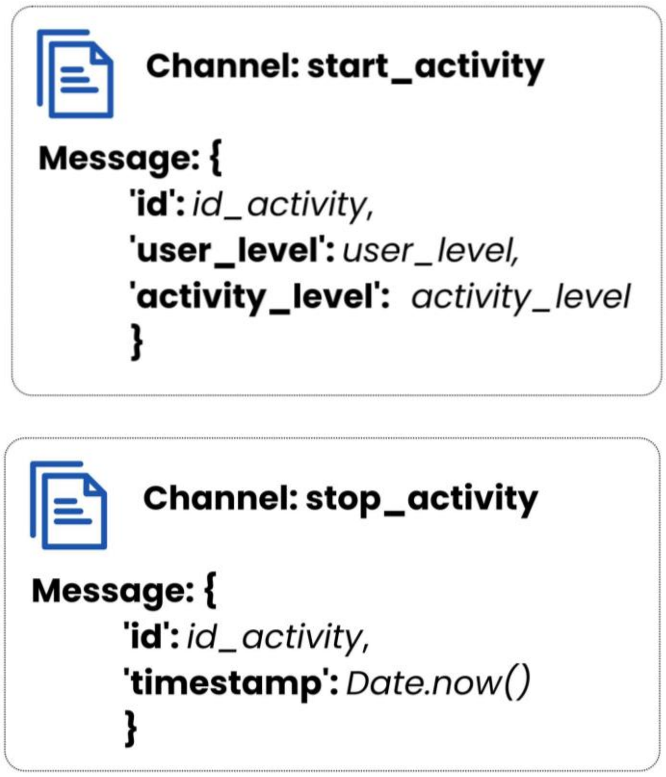
The UIExample.cs script contains a script with a simple example of how to connect with Redis as both a publisher and subscriber and how to translate the Redis messages into Unity internal variables.

Message Formats

The message formats exchanged by the VR application and the Personalization Tool are as follows:

[15] https://github.com/GlitchEnzo/NuGetForUnity?tab=readme-ov-file#unity-20193-or-newer

1. Unity application as Publisher, i.e., sending messages (two channels, Figure 30):



Channel: start_activity

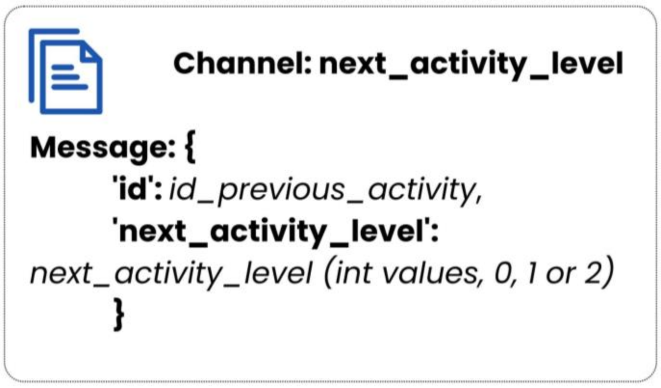
Message: {
    'id': *id_activity,*
    'user_level': *user_level,*
    'activity_level': *activity_level*
    }

Channel: stop_activity

Message: {
    'id': *id_activity,*
    'timestamp': *Date.now()*
    }

Figure 30. Messages format the Personalization Integration Template sends when communicating with the Personalization Tool, 1) message indicating the start of activity (top), 2) message indicating the end of activity (bottom).

2. Unity application as Subscriber, i.e., receiving messages (one channel, Figure 31):



Channel: next_activity_level

Message: {
    'id': *id_previous_activity,*
    'next_activity_level':
next_activity_level (int values, 0, 1 or 2)
    }

Figure 31. Message format the Personalization Integration Template expects to receive from the Personalization Tool.

Important Point

Running complex unity code in the method used for handling the subscribed Redis messages can be complicated when debugging in Unity - (e.g., ProcessMessageNextActivityLevel in the UIExample.cs).

It is best for the methods that handle the subscribed messages to only update primitive variables and have the other standard Unity method update more complex components based on the primitive variables just populated.

5.2. INTEGRATION OF MAGIC XROOM AND PERSONALIZATION ENABLERS

To illustrate the integration of Personalization Enablers and VR applications with the capability to collect user data, we implemented a demonstration using Magic XRoom. In this demonstration, we integrated Magic XRoom (a VR application with the functionality of collecting and recording users' data through sensors) with Personalization Enablers, achieving near real-time latency.

The demonstration setup involves two machines, as shown in Figure 32:

- Machine 1 (Windows OS): This machine runs the Magic XRoom application and deploys the *Inference Preprocessing component* (per modality) from *Inference Tools* (refer to Section 3.3.2.1). It is essential for this component to be deployed in the same machine as Magic XRoom, as it reads data files recorded by Magic XRoom.
- Machine 2 (MacOS, Linux or Windows OS): This machine is responsible for deploying all the other Personalization Enablers for Inference, Personalization Tool and Personalization Dashboard. These components could be deployed in Machine 1; however, it would not be practical because the computational processing power required by Magic XRoom would leave other components without the resources they need to function.

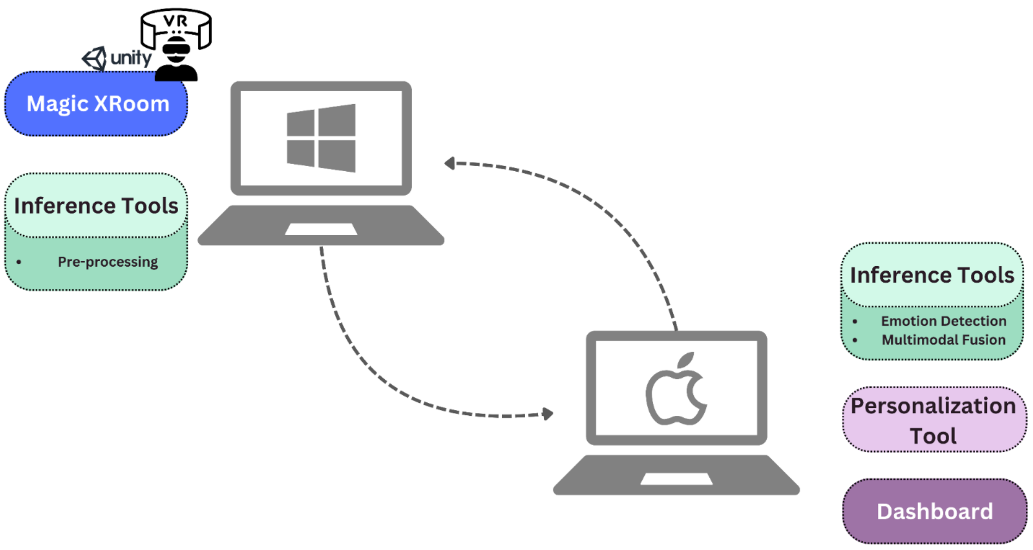


Figure 32. Hardware setup with two machines for the demonstration of the integration of Magic XRoom and the Personalization Enablers. These machines are connected to the same internal network.

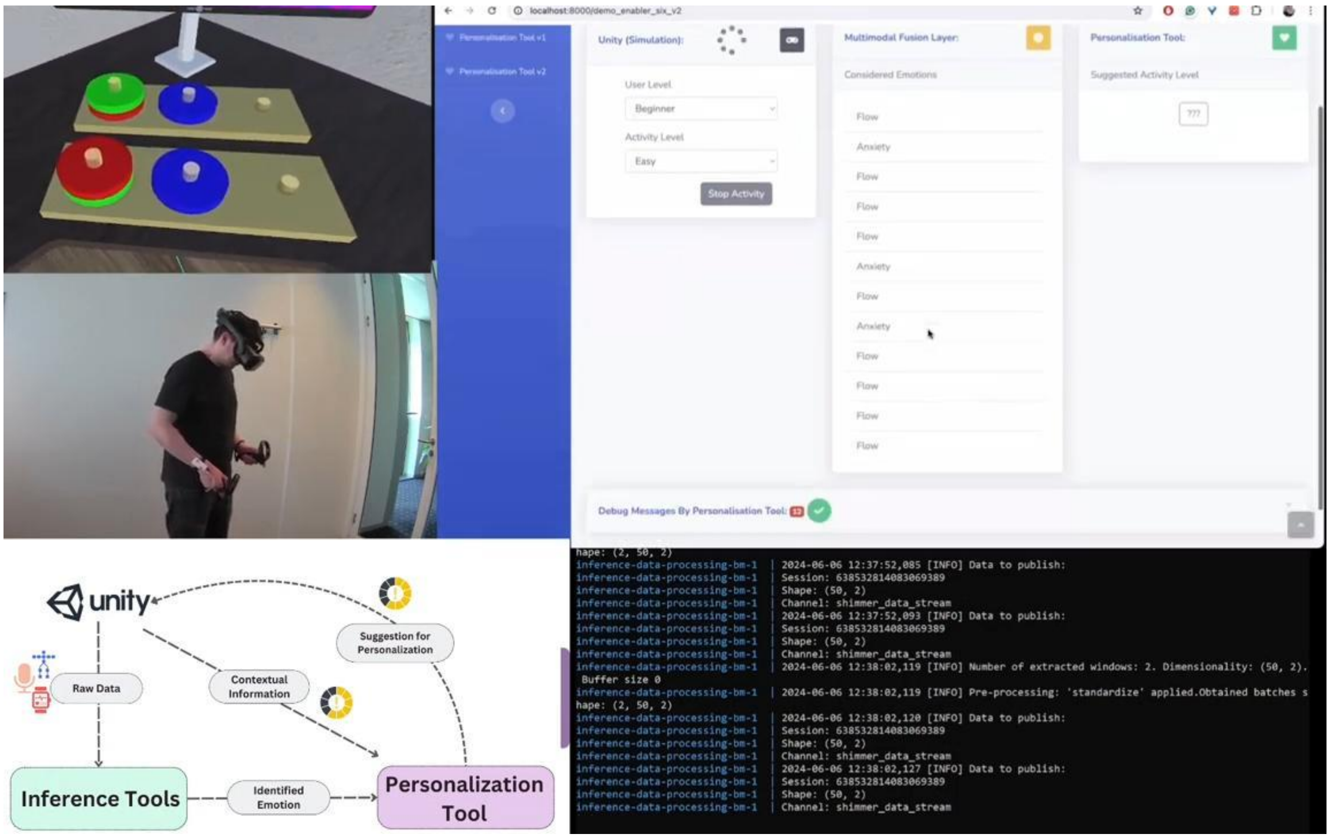
During the demonstration, the user engages with Magic XRoom while using Shimmer sensors. Magic XRoom collects and records the multimodal sensor data in near real-time. For this demonstration, we focused on two modalities: *bio-measurements* and *body-tracking*. The *inference preprocessing component* reads the data generated by Magic XRoom, processes it into features, and organizes these features into temporal windows. Each feature's time window is then sent as a message to the *emotion classification component* for each modality (as described in Section 3.3.2.1) .

The *emotion classification component* utilizes a trained machine learning model (trained using the Training Tools in Section 3.2) to classify the user's engagement into three emotional states: *engagement, boredom, or frustration*. Subsequently, this classification is sent as a message to the *multimodal fusion component*.

The multimodal fusion component receives the user's engagement classification from different modalities, organizes and matches them to the corresponding time window (Section 3.3.2.2), and then executes the multimodal fusion, generating a single multimodal prediction for each given time window. For *bio-measurements* and *body tracking* modalities specifically, each time window spans five seconds. Finally, the *multimodal fusion component* sends the fused calculated prediction to the Personalization Tool.

Lastly, the Personalization Tool receives messages from the *multimodal fusion component* and combines them with the *user and activity challenge levels* information, using them to compute personalized suggestions in the form of *activity challenge levels*.

The multimodal predicted emotions, the user and activity challenge levels, and the Personalization Tool suggestions can then be visualized in real time through the graphical interface of the Personalization Dashboard. Figure 33 illustrates this process in the demonstration. In this demonstration, we utilized the Personalization Dashboard to simulate the Unity application input, i.e., user and activity challenge levels information, since Magic XRoom does not include this functionality, and the update of BA1, as described in Section 5.4, is still in progress.



```
hape: (2, 50, 2)
inference-data-processing-bm-1  | 2024-06-06 12:37:52,085 [INFO] Data to publish:
inference-data-processing-bm-1  | Session: 638532814083069389
inference-data-processing-bm-1  | Shape: (50, 2)
inference-data-processing-bm-1  | Channel: shimmer_data_stream
inference-data-processing-bm-1  | 2024-06-06 12:37:52,093 [INFO] Data to publish:
inference-data-processing-bm-1  | Session: 638532814083069389
inference-data-processing-bm-1  | Shape: (50, 2)
inference-data-processing-bm-1  | Channel: shimmer_data_stream
inference-data-processing-bm-1  | 2024-06-06 12:38:02,119 [INFO] Number of extracted windows: 2. Dimensionality: (50, 2).
 Buffer size 0
inference-data-processing-bm-1  | 2024-06-06 12:38:02,119 [INFO] Pre-processing: 'standardize' applied.Obtained batches s
hape: (2, 50, 2)
inference-data-processing-bm-1  | 2024-06-06 12:38:02,120 [INFO] Data to publish:
inference-data-processing-bm-1  | Session: 638532814083069389
inference-data-processing-bm-1  | Shape: (50, 2)
inference-data-processing-bm-1  | Channel: shimmer_data_stream
inference-data-processing-bm-1  | 2024-06-06 12:38:02,127 [INFO] Data to publish:
inference-data-processing-bm-1  | Session: 638532814083069389
inference-data-processing-bm-1  | Shape: (50, 2)
inference-data-processing-bm-1  | Channel: shimmer_data_stream
```

Figure 33. Integration of Magic XRoom and the Personalization Enablers. The top left shows the user playing Magic XRoom while wearing sensors and the user's view. The bottom left shows a diagram of the information exchanged by the components in the demonstration. The top right shows the Personalization Dashboard (as illustrated in Figure 15) . The bottom right shows a command terminal with the inference preprocessing running in the Windows machine.

This demonstration showcases the near real-time capabilities of the Personalization Enablers in processing multimodal input data to identify user engagement levels. Combining this information with additional context—such as user levels and activity challenge levels—generates personalized suggestions that can be implemented by the VR application to adapt the user's learning experience automatically. It also highlights the robustness of the Personalization Enablers' software architecture, which allows different components to be deployed on multiple machines and communicate with one another through a network. This flexibility is essential for managing and ensuring the hardware requirements of each enabler or tool.

5.3. DATA COLLECTION MODULE/PLUGIN

5.3.1. Description

The Data Collection Module is a software plugin currently under development, designed to configure and manage sensors and devices for data collection purposes, as shown in Figure 34. Extracted from a larger application (Magic XRoom), it is being restructured to function as a standalone tool to boost its adoption by third parties. The module will support the collection of various types of data and save the results in separate CSV files for seamless storage and accessibility. This modular approach is intended to simplify integration into different systems and streamline data handling workflows.

Incorporating this standalone module into a VR Unity application is an essential step for accessing the Personalization Enablers functionality since the Personalization Enablers require this collected data as input to process and produce personalized suggestions for adaptation.

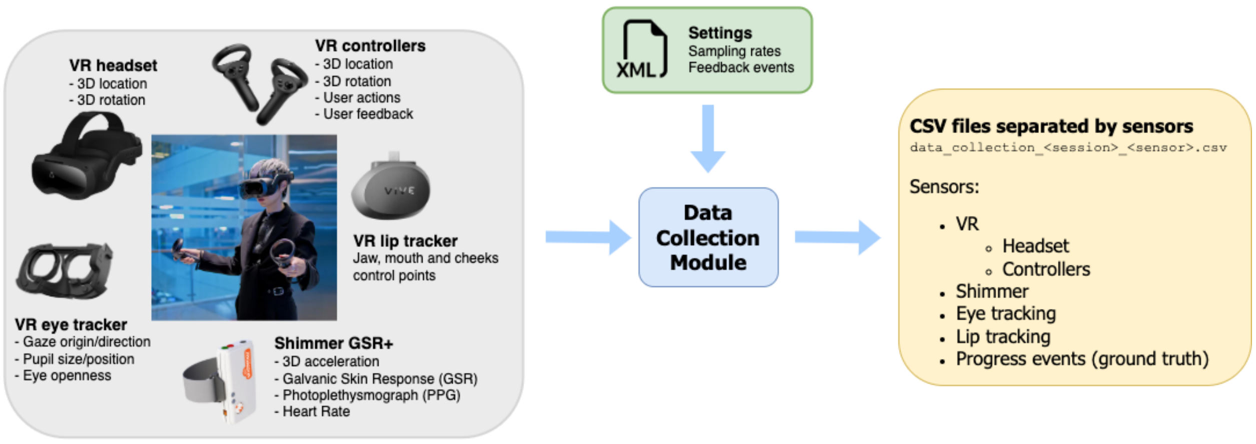


Figure 34. Data Collection module workflow

5.3.2. Prerequisites

The Data Collection Module will be distributed initially as a standalone C# library, requiring a development environment compatible with .NET. This version will allow integration into custom C# applications with minimal setup. In the future, the module will also be available as a Unity package, enabling seamless integration with Unity projects.

To use the module, users should have a basic understanding of C# programming and, for the Unity package, familiarity with Unity's development environment and asset management.

5.3.3. Installation

The Data Collection Module will be available for installation in two formats. Initially, it will be distributed as a standalone C# library, which can be added to a project by referencing the library in the development environment. The documentation will provide instructions for downloading and including the library, along with any required dependencies.

In its later release as a Unity package, installation will involve importing the package directly into a Unity project via Unity's Package Manager or manually importing it. Detailed steps for integrating the package, along with any configuration settings, will be outlined to ensure smooth implementation.

5.4. FUTURE WORK IN INTEGRATION OF BEACON APPLICATIONS WITH PERSONALIZATION ENABLERS

The challenge of integrating Beacon Applications and Personalization Enablers was divided into different sub-tasks, which progress was reported in the previous subsections. To facilitate this process for both XR2Learn partners and third parties, we:

1. have created a Unity Integration Template which includes all the main functionalities and the required communication channels and its documentation have been released,
2. showcased the integration of Magic XRoom with Personalization Enablers in a real-time setup has been showcased, and
3. are finalizing the creation of the data collection modules from Magic XRoom as a standalone tool.

It is worth mentioning that each of these steps has been tested and validated individually and successfully integrated with additional parts as a proof of concept to demonstrate their feasibility for use by third parties.

As a further step to demonstrate a full integration of Personalization Enablers and Beacon Applications, we will utilize *Beacon Application 1: Laser Cutting Machine* as a showcase. This Beacon Application will be updated to a new version, which will include the data collection modules extracted from Magic XRoom and implement the Unity Integration Template to communicate with the Personalization Enablers.

Beacon Application 1 will additionally extend its functionality to automatically adapt the educational content according to the suggestions received from the Personalization Enablers. Some examples of content adaptation could be presenting more hints to the user, and skipping or repeating challenging levels, among others. The Beacon Application 1 update work is planned to commence within the first quarter of 2025.

6. CONCLUSION

This document describes the progress achieved in Task 3.2, "XR2Learn Enablers", during the months 14 to 26 of the XR2Learn project, which is part of XR2Learn Phase B, the second sub-phase as described in Section 1.2.2 of the proposal document. Task 3.2 focuses on designing, implementing, and delivering novel enablers for XR applications. During the second sub-phase, work on improvements to the enablers and the integration of enablers and beacon applications has been conducted.

The enablers' improvements involve adding a new modality, multimodal capacity, low latency (near real-time) Inference and Personalization, more sophisticated Personalization heuristics, integration with data collection tool (together with new data collection conducted), new trained ML models available, and documentation extension, among others. The report also includes information on additional enablers developed beyond the project proposal. These enablers have been updated, and new ones have been created. These include **Magic XRoom** (a data collection tool), a **command line interface** to make the enablers more user-friendly, the **personalization enabler template** (a tool to automate and facilitate creating new personalization enablers), and a graphical user interface **dashboard** to demonstrate, debug and facilitate the communication between Personalization Tool (Enabler 6), Inference Tools, and XR Unity applications.

As to lay the foundations of *integration* between enablers and beacon applications, two sub-tasks have been conducted in Task 3.2:

1) The **data collection modules** from Magic XRoom have been validated in real-life scenarios and are in the process of being extracted and converted into an independent module that can be re-utilized within a beacon application, extending its functionality to collect user data.

2) A new enabler was created, **Personalization Integration Template**, a Unity application to showcase and facilitate the integration process between a Unity application with the Personalization Enablers.

Additionally, we have demonstrated the integration of Personalization Enablers and Magic XRoom, showcasing their near real-time processing capability.

As the last step in completing the integration of enablers and beacon applications, *Beacon Application 1, the laser-cutting machine,* will be extended to include the two sub-tasks described above as a proof of concept. This update on Beacon Application 1 is planned to commence in the first quarter of 2025.

All tools described were developed following established software engineering practices to foster principles of open science, such as reproducibility, open access, transparency, software sustainability, and quality. As listed in Table 2, they are also available as open-source repositories on GitHub.

Lastly, we plan to start writing for a scientific publication featuring the Personalization Enablers in March 2025. We aim to publish it in a relevant Q1 scientific journal, which will be determined in the coming months.