Ref. Ares(2026)7006203 - 14/07/2026

HUMAN-CENTRIC AI-ENABLED EXTENDED REALITY APPLICATIONS FOR THE INDUSTRY 5.0 ERA

D3.4 – PERSONALISED XR CONTENT DEVELOPMENT AND ADAPTATION V2

Lead Beneficiary	UNP
Work Package Ref.	WP3 – Human Centric Digital Twins and XR Personalisation
Task Ref.	T3.4 – Personalised Content Development T3.5 – XR Content Adaptation and Personalisation Mechanisms
Deliverable Title	D3.4 – Personalised XR Content Development and Adaptation v2
Due Date	2026-06-30
Delivered Date	2026-07-14
Revision Number	3.0
Dissemination Level	Public (PU)
Type	DEM
Document Status	Release
Review Status	Internally Reviewed and Quality Assurance Reviewed
Document Acceptance	WP Leader Accepted and Coordinator Accepted
EC Project Officer	Ms. Marta PALAU FRANCO

HORIZON-CL4-2023 Research and Innovation Action

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement no 101135209.

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

DISCLAIMER

The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Union. Neither the EASME nor the European Commission is responsible for any use that may be made of the information contained therein.

COPYRIGHT MESSAGE

This report, if not confidential, is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0); a copy is available here: https://creativecommons.org/licenses/by/4.0/. You are free to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon the material for any purpose, even commercially) under the following terms: (i) attribution (you must give appropriate credit, provide a link to the license, and indicate if changes were made; you may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use); (ii) no additional restrictions (you may not apply legal terms or technological measures that legally restrict others from doing anything the license permits).

CONTRIBUTING PARTNERS

Partner Acronym	Role[1]	Name Surname[2]
IML	Contributor	Jorge Oliveira and Tomaz Saraiva
SUPSI	T3.5 Leader	Sara Masiero and Elias Montini
UNP	T3.4 Leader	Joana Guedes and Bruno Almeida
UBI	Internal Reviewer	Nikolaos Tousert
LNS	Internal Reviewer	Saurer Philipp
UPRC	Quality Assurance	Thanos Kiourtis, Argyro Mavrogiorgou

REVISION HISTORY

Version	**Date**	**Partner(s)**	**Description**
0.1	2026-04-28	UNP	1st ToC Version
0.2	2026-05-15	SUPSI	Add Chapter 2 and 3
0.3	2026-05-26	SUPSI	Add Chapter 4
0.4	2026-06-09	UNP	Add Chapter 5
0.5	2026-06-18	IML	Updated Chapter 5
0.6	2026-06-22	UNP	Updated ToC
0.7	2026-06-25	SUPSI	Updated Chapter 3
1.0	2026-07-02	UNP	1st Version
1.1	2026-07-07	UNP	Version for Peer Reviews
2.0	2026-07-10	UNP	Version for Quality Assurance
3.0	2026-07-10	UPRC, GFT	Version for Submission

[1] Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance
[2] Can be left void

LIST OF ABBREVIATIONS

Acronym	Definition
AAS	Asset Administration Shell
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
AUI	Adaptive User Interface
DTLS	Datagram Transport Layer Security
GDPR	General Data Protection Regulation
GenAI	Generative Artificial Intelligence
HCI	Human Computer Interaction
HDTs	Human-Centric Digital Twins
HMI	Human Machine Interface
I5.0	Industry 5.0
ID	Identification
IdP	Identity Provider
iOS	iPhone Operating System
JSON	JavaScript Object Notation
JWT	JSON Web Token
LLM	Large Language Model
MR	Mixed Reality
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
QR	Quick Response
RBAC	Role-Based Access Control
REST	Representational State Transfer
SAS	Stress Analog Scale
SDK	Software Development Kit
SP	Service Provider
SRTP	Secure Real-Time Transport Protocol
SSO	Single Sign-On
SSQ	Simulator Sickness Questionnaire
TLX	Task Load Index
TPAT	Training Programs Authoring Tool
UC	Use Case
UI	User Interface
UUID	Universal Unique Identifier
UX	User Experience
UXQ	User Experience Questionnaire
VR	Virtual Reality
WebRTC	Web Real-Time Communication
WP	Work Package
XR	Extended Reality
XRTP	XR Training Plugin

EXECUTIVE SUMMARY

Deliverable D3.4 – *Personalised XR Content Development and Adaptation v2* – is developed within Work Package 3 (WP3) of the XR5.0 project, which focuses on enabling human-centric digital twins and personalised extended reality (XR) experiences in the context of Industry 5.0.

The main objective of this deliverable is to design, develop, and validate methods and technologies for personalised XR content and adaptive training systems. The work consolidates and extends the foundations established in the previous version (D3.3), moving from conceptual frameworks and rule-based adaptation towards integrated, AI-driven and operational solutions deployed within the XR5.0 ecosystem.

The approach adopted in this deliverable combines three complementary dimensions. First, a refined framework for Adaptive User Interface (AUI) design is introduced, providing a structured methodology for developing human-centric adaptive interfaces. Second, advanced personalisation mechanisms are developed based on behavioural data and Generative Artificial Intelligence (GenAI), enabling both implicit adaptation (driven by user interactions) and explicit customisation (based on user preferences). Third, these adaptation capabilities are operationalised through the Adaptation Engine, a service-oriented component that integrates with the XR Training Platform and the Clawdite human-centric digital twin infrastructure.

The deliverable further demonstrates the integration of these components in real XR environments, including the XR Training Plugin and Pilot 4 Virtual Reality application. Key functionalities such as stress and workload monitoring, real-time content adaptation, and user authentication are implemented and interconnected, ensuring a seamless flow of data between users, XR applications, and the digital twin.

The main outcomes of this work include: (i) a validated framework for adaptive XR content design; (ii) a GenAI-powered adaptive training system capable of delivering personalised experiences in real time; (iii) a fully functional Adaptation Engine supporting scalable and interoperable integration; and (iv) the successful deployment of these components within XR5.0 pilot scenarios.

Overall, D3.4 demonstrates the feasibility and effectiveness of combining human-centric design principles, AI-driven adaptation, and XR technologies to support personalised training and interaction in industrial environments, contributing significantly to the XR5.0 vision of Industry 5.0.

TABLE OF CONTENTS

1 Introduction 8
1.1 Objectives of the Deliverable 8
1.2 Insights from Other Tasks and Deliverables 8
1.3 Updates from Previous Version 8
1.4 Structure of the Deliverable 9
2 Framework for Adaptative XR Content Design 10
2.1 Previous work 10
2.2 Updates and improvements 10
3 Algorithms for Adaptive Training 12
3.1 Human-Centric Behaviour Driven Personalization 12
3.1.1 Personalization Algorithms powered by GenAI 12
3.1.2 Model Evaluation 17
3.1.3 Summary 22
3.1.4 Main Outcomes 23
4 Clawdite and the Adaptation Engine for the Training Platform 24
4.1 From the UI Adaptation Engine Proof of Concept to a Content-Aware Service 24
4.2 Design Rationale 25
4.3 Personalisation Operations and the Context Model 25
4.3.1 The Context Object 25
4.3.2 Competency Gating of the Learning Path 26
4.4 Data Model: Materials and Adaptation Decisions 26
4.5 Adaptation Strategies and the Deterministic Safety Net 27
4.6 Service Interface and Division of Responsibilities 28
4.7 Integration with Clawdite and the Training Platform 28
4.8 Future Work and Integrations 29
5 Integration in XR Training Platform and Digital Twin 30
5.1 Mental Workload and Stress Questionnaire Integration in VR Environment 30
5.2 Adaptive Content Personalisation in the XR Training Platform 32
5.3 Authentication and User Identification Solution 33
5.3.1 Overview: the Identity Bridge 34
5.3.2 Architecture and Trust Boundary 34
5.3.3 Authentication and Resolution Flow 35
5.3.4 Identity-Mapping Mechanism 36
6 Conclusions 37
6.1 Lessons Learned 37

LIST OF FIGURES

Figure 1 – The Framework for AUI Design: Updated Version 11
Figure 2 – Internal Architecture of the Implicit and Explicit Personalized Mechanism Feeding the GENAI Reasoning Engine 13
Figure 3 – Onboarding Form for Explicit Customization: Users provide language preferences, prior experience, learning objectives, usability requirements, and additional requests before commencing the training session 17
Figure 4 – Screenshot as an Illustrative Example of the Adaptive Model Interface at Step 16 with Notes on Component Details and Explanations 18
Figure 5 – Illustrative Example of the UI for User ID 5 at Step 4. The screenshot shows the content format selections generated by the system in response to the user's explicit preferences and requests 18
Figure 6 – Representation of the Implicit User Data Exploited by the System, together with an Explanation of the Adaptive Reasoning Process and the Decisions Adopted by Gemini to Personalize Recommendations and Content Configurations 19
Figure 7 – Adaptation Engine within the XR5.0 Architecture 24
Figure 8 – XR Platform and HDT Integration Architecture 30
Figure 9 – Mental Workload and Psychological Stress Evaluation Panel in the Pilot 4 VR Application 31
Figure 10 – Stress Index Calculator within the Clawdite Human-Centric Digital Twin: High-Level Functional View 32
Figure 11 – Static Customisation and Dynamic Adaptation Loop in the XR5.0 Training Platform, driven by HDT Data and User Behaviour 33
Figure 12 – Identity Bridge Architecture and Trust Boundary between Client Applications and the HDT Backend 35
Figure 13 – Authentication and Identify-Resolution Flow linking XR Applications to the HDT 36

LIST OF TABLES

Table 1 – Overall Format-Prediction Accuracy under LOO-CV 21
Table 2 – Prediction Accuracy by Step Type 21
Table 3 – Performance by Institutional Format 22
Table 4 – Benchmark: Dynamic Rule-Based VS GenAI-Powered 23
Table 5 – The Six Supported Material Types and the Elements the Engine can Adapt 26
Table 6 – Tunable Parameters of the Deterministic rule_based_v1 Strategy 27
Table 7 – REST Interface of the Adaption Engine 28
Table 8 – Allocation of Identity Responsibilities between Hololight Hub and Clawdite 34

LIST OF CODES

Code 1 – Structure of the System Prompt 15
Code 2 – JSON Schema of the Structured Output Returned by the Adaptation Model 15

1 INTRODUCTION

The XR5.0 project aims to develop human-centric, AI-enabled XR solutions to enhance industrial processes in the context of Industry 5.0. Within this framework, WP3 focuses on the design and implementation of Human-Centric Digital Twins (HDTs) and the development of personalised XR content and adaptive interaction mechanisms.

Deliverable D3.4 – *Personalised XR Content Development and Adaptation v2* – represents the second iteration of the work carried out under Tasks T3.4 and T3.5. Building upon the foundations established in Deliverable D3.3, this deliverable advances the state of the art by transitioning from conceptual and rule-based approaches to fully implemented, AI-driven adaptive systems integrated within the XR5.0 platform.

The objective of this deliverable is to define, implement, and validate mechanisms for generating and adapting XR content based on individual user characteristics, behavioural data, and contextual information derived from the human digital twin. The work aims to enable dynamic and personalised training experiences that adapt in real time to users' skills, preferences, cognitive load, and emotional state.

The deliverable adopts a multi-layered methodology. At the design level, it refines the AUI framework to support structured and human-centric adaptation strategies. At the algorithmic level, it introduces a novel adaptation approach based on GenAI, combining implicit behavioural modelling with explicit user-driven customisation. At the system level, it operationalises these mechanisms through the Adaptation Engine, which serves as the core component for delivering personalised content to XR applications.

In addition, the deliverable demonstrates the integration of these technologies within the broader XR5.0 ecosystem, including the XR Training Platform, the Clawdite human-centric digital twin, and pilot XR applications. This integration enables real-time data exchange and coordinates adaptation across multiple components, ensuring consistency and scalability of the personalised experience.

1.1 Objectives of the Deliverable

The main objectives of Deliverable D3.4 are:

- To extend and refine the framework for personalised XR content development;
- To design and implement adaptive mechanisms for XR content personalisation;
- To develop AI-driven algorithms for behaviour-based and user-driven adaptation;
- To integrate adaptive functionalities within the XR Training Platform and digital twin infrastructure;
- To validate the proposed solutions in realistic XR training scenarios.

1.2 Insights from Other Tasks and Deliverables

This deliverable builds upon and integrates results from multiple XR5.0 activities:

- D3.2: Human-centric Digital Twin design and stress modelling;
- D3.3: Initial AUI framework and rule-based adaptation;
- WP5: XR Training Platform and XR Training Plugin;
- Pilot activities: Validation in industrial XR environments.

1.3 Updates from Previous Version

Compared to Deliverable D3.3, this version introduces several key advancements:

- Transition from rule-based to GenAI-powered adaptation;
- Integration of implicit and explicit personalisation mechanisms;
- Development of a fully operational Adaptation Engine;
- Deployment in real XR environments and pilot scenarios;
- Enhanced integration with the human digital twin (Clawdite).

1.4 Structure of the Deliverable

This deliverable is structured as follows:

- Chapter 2 presents the updated framework for adaptive XR content design;
- Chapter 3 introduces the human-centric adaptive algorithms and GenAI-based mechanisms;
- Chapter 4 describes the Adaptation Engine and its architecture;
- Chapter 5 details the integration within the XR Training Platform and digital twin ecosystem;
- Chapter 6 summarises the conclusions and lessons learned.

2 FRAMEWORK FOR ADAPTATIVE XR CONTENT DESIGN

2.1 Previous work

The previous work of this deliverable D3.3 laid the foundations for the development of AUIs. It established both the methodological and technical basis for adaptation by defining a structured seven-step protocol, as well as the supporting framework and associated design components required for implementing adaptive interfaces. The protocol provides a systematic blueprint for developers and designers, guiding them through the key stages of AUI development: from static interface prototyping and empirical testing, through interaction data analysis and adaptive logic specification, to final implementation, evaluation, and iterative refinement. Alongside the protocol, a framework was introduced to structure adaptation design around three core dimensions: i) adaptation initiators (user-driven, system-driven, and hybrid), ii) adaptation targets (content reordering, highlighting, information folding, multimedia enhancements), and iii) adaptation features (layout adaptation, task feature minimization/maximization, navigation structure).
The protocol was validated through an experimental case study in which participants assembled a LEGO® forklift following a 16-step procedure encompassing three operation types: component picking, assembly, [1] and quality control. A static interface was first tested with ten participants to collect interaction logs and identify behavioural patterns.
Based on this analysis, two adaptive modes were defined and implemented, a Pattern-Guided Mode and a Preference-Adaptive Mode and subsequently evaluated against a static baseline. Results indicated that adaptive modes reduced the number of user interactions and suggestion requests, lowered error rates, and improved subjective ratings of usefulness, clarity, and user satisfaction compared to the static interface. The full details of this work, including the methodology, experimental setup, and results, are reported in "A Protocol for Human-Centric Adaptive User Interfaces: From Static Interaction to Behaviour-Driven Adaptation paper" [1] Based on these findings, several directions were identified for potential future development:

- Integration of user-level variables, such as skill level, cognitive load, and past interaction history, into the adaptation logic;
- Introduction of user clustering to group participants with similar behavioural profiles and enable more targeted personalization;
- Development of machine learning models or recommendation systems capable of providing tailored content suggestions;
- Further validation through A/B testing and additional experimental iterations with more diverse user populations.

Prior to advancing towards these objectives and developing new system components, a revision of the work conducted so far was carried out. The results of the review activities are presented in the following section 2.2*. Updates and improvements*.

2.2 Updates and improvements

Beginning with Step 7 of the protocol – "Review, Refinement, and Iteration" – this section reports the outcomes of a structured revision of the framework for AUI design presented in the previous deliverable D3.3. The revision process leveraged the authors' extensive expertise, complemented by structured peer reviews, critical reflections and collaborative meetings. Although the updates are minor and only concern the framework, they were necessary to eliminate ambiguities and inconsistencies that could hinder its readability and practical application. The updated framework is illustrated in Figure 1.
The following improvements were introduced:

- **Terminology clarification in the "How?" block.** The label "Adaptation Target" was renamed to "Adaptation Mechanism". The term *target* could be interpreted as referring to goals or objectives, which does not reflect the purpose of this block. Since the block focuses on the ways information is presented in the UI, *mechanism* was considered a clearer and more accurate term.

- **Removal of redundant elements.** Within the “What?” block, the item "Task Feature Minimization/Maximization", defined as the adjustment of the amount and complexity of information presented based on current context, was removed. This concept was found to be redundant with respect to the "Folding" mechanism already present in the "How?" block, and its removal contributes to a leaner and less repetitive framework structure.
- **Layout improvements.** The visual layout of the framework blocks was redesigned to support a more intuitive and straightforward interpretation, particularly for practitioners and researchers interested in designing or developing adaptive interfaces.

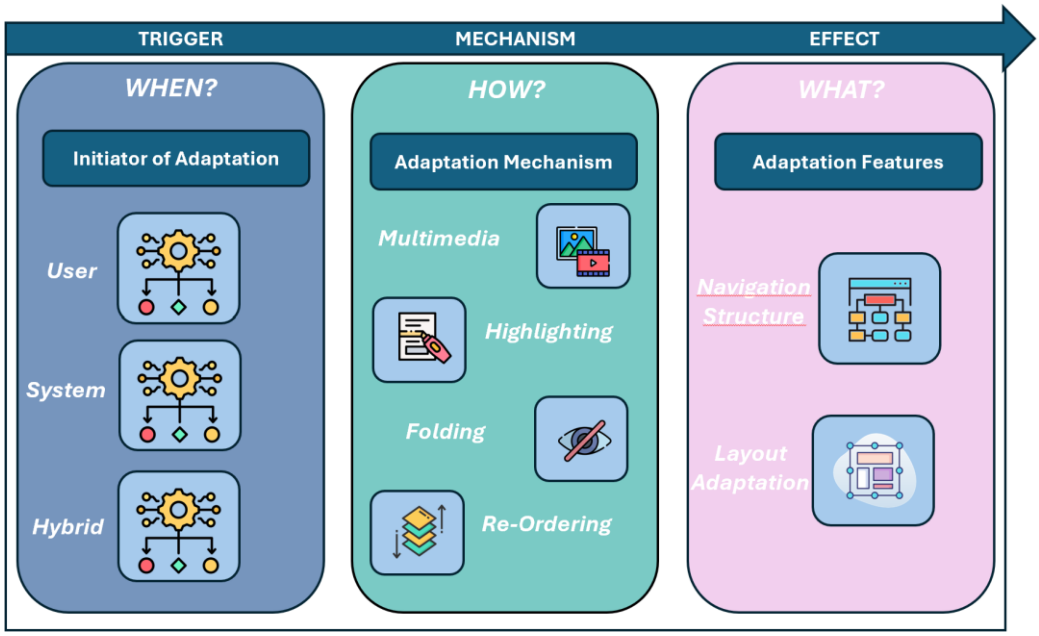


Figure 1 – The Framework for AUI Design: Updated Version

3 ALGORITHMS FOR ADAPTIVE TRAINING

3.1 Human-Centric Behaviour Driven Personalization

Effective adaptive training support in manufacturing environments requires more than the delivery of structured information, it demands a dynamic adaptation of instructional content to the individual characteristics and real-time behavioural signals of each operator [2]. By tailoring content modality, complexity, and format to the user, AUIs hold the potential to reduce cognitive load, accelerate skill acquisition, and improve overall training effectiveness. However, this objective introduces significant design challenges: development teams often lack sufficient prior knowledge of target users' interaction preferences, calibration processes must be repeated across different deployment contexts, and the literature highlights a persistent lack of consensus on appropriate threshold values across adaptation components [3].

This challenge is further compounded by the evolving demographic composition of the European manufacturing workforce. According to EIT Manufacturing[3], 32% of the 32 million European manufacturing workers are currently aged between 50 and 64, with mass retirements projected over the next 10 to 15 years. However, only about one third of companies have analysed their workforce demographics and projected the impact of retirements, while the remaining two thirds are planning future capacity on the assumption that current productivity levels will hold, which it will not. As a consequence, adaptive systems must be designed to serve increasingly heterogeneous user profiles: from digitally fluent younger workers to experienced older operators with different relationships to technology and instruction formats. For AUI design, this means that adaptive logic must go beyond visual parameters and address the modality, language, and content density of instructional information to support knowledge transfer across a multigenerational workforce.

Building on the objectives identified from the previous work, as the integration of user-level variables into the adaptation logic, and the introduction of user clustering to group participants with similar behavioural profiles, it was deemed necessary to extend the existing adaptive system with mechanisms that address both user personalization and customization.

To this end, **human-driven triggers** were introduced as a core extension of the adaptation logic, making the system truly **human-centric** by enabling adaptations that respond not only to observed interaction patterns but also to explicit user characteristics, preferences, and requirements. This implies moving beyond the analysis of interaction patterns and behavioural signals alone, to incorporate user profiling dimensions such as skill level, language preference, individual preferences, and explicit user intent. The following sections describe how these human aspects were integrated through the development of a new adaptation logic system, powered by GenAI.

3.1.1 Personalization Algorithms powered by GenAI

Recent advancements in AI, particularly in the field of GenAI, a subset of AI that is able to create original content such as text, images, video, audio or software code in response to a user's prompt or request[4], and large language models (LLMs), mark the beginning of a new generation of innovations in UI and User Experience (UX) design [4] [5]. These technologies offer a promising shift towards more personalised and adaptable design processes, paving the way for solutions to challenges in AUI development.

Unlike previous AI-driven methods that relied on predefined user profiles, LLMs can interpret user behaviour and contextual information in real-time, allowing the system to generate context-aware and dynamically tailored interactions. As a result, the integration of LLMs into industrial Human Machine Interface (HMI) is becoming an important and promising research area for enabling user-centred adaptation in interactive systems [6].

[3]https://www.eitmanufacturing.eu/wp-content/uploads/2026/02/eit-manufacturing-industrial-innovation-report-2026_27022026.pdf

[4] https://www.ibm.com/think/topics/generative-ai

Building on this potential, GenAI-based approach was integrated into the adaptive system to address the human-centricity objectives. The manually configured, rule-based adaptation logic developed in previous work was replaced by an LLM-based solution using the Google Gemini model and an in-context learning approach.

The resulting system adopts a **dual-layer adaptation architecture** that combines real-time behavioural data with direct user input. Adaptation decisions are generated autonomously by the AI agent using two complementary information sources:

- **Implicit personalization,** based on user behaviour and performance data, allowing the system to adapt the interface automatically;
- **Explicit customization**, where users can express their preferences or requirements through natural language prompts, enabling on-demand interface reconfiguration.

Figure 2 illustrates how these two information sources are structured and converge into the reasoning engine, and how the resulting adaptation decision is applied to the AUI.

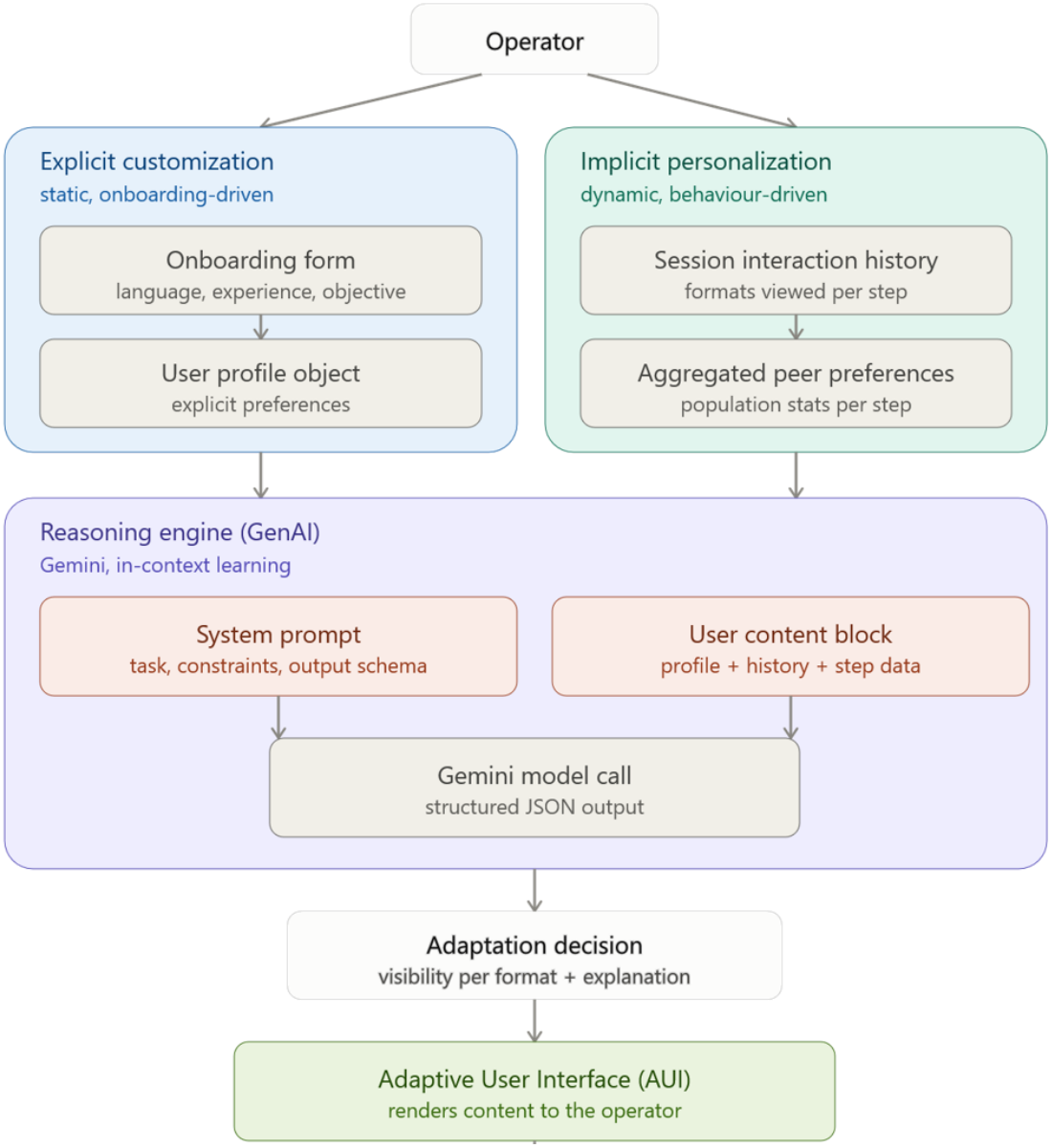


Figure 2 – Internal Architecture of the Implicit and Explicit Personalized Mechanism Feeding the GENAI Reasoning Engine

To operationalize these mechanisms, the adaptation process relies on a prompt-based architecture composed of two main components. The first is a developer-defined system prompt, which specifies the task context, available content formats, adaptation logic, and the expected output structure. The second is a dynamically generated user input block that aggregates behavioural metrics, user preferences, interaction history, and task-specific information collected throughout the session. Together, these components constitute the complete prompt provided to the LLM at each adaptation cycle.
Based on this information, the model generates a structured JSON output that specifies which content formats should be displayed and provides a natural-language explanation of the adaptation decision. To ensure reliable integration with the adaptive interface, the model output is constrained through a predefined JSON schema using Gemini's structured output capabilities. Code 1outlines the structure of the system prompt that defines the adaptation logic. Code 2 shows the JSON schema enforced on the model output through Gemini's structured output feature.
The model temperature was set to 0.7, providing a balance between consistency and flexibility. This configuration allows the system to generate coherent and repeatable adaptation decisions while remaining responsive to different user behaviours, preferences, and task contexts.

```
SYSTEM PROMPT — structure
_____________________________________

System = (
TASK DEFINITION
  "Adapt the visibility of five UI elements per training step:
   short_text, long_text, image_single_pieces, image_assembly, video."

HARD CONSTRAINTS
  - short_text and long_text are mutually exclusive (never both true)
  - visibility can only be true if the corresponding media path is non-empty
  - media file paths must never be altered or invented

STEP-TYPE CONSTRAINTS
  - withdraw:         only short_text, single_pieces available
  - quality_control:   no image_assembly available
  - assembly:          all content types available

ADAPTATION LOGIC
  - infer user experience level from free-text profile (novice / expert)
  - infer visual vs. analytical learning style from stated goals
  - infer readability needs from reported screen setup
  - weigh the user's own interaction history slightly more than
    the aggregated peer preferences for the same step type
  - treat the first 1-2 occurrences of a step type as low-reliability
    signal (cold start); increase confidence from the 3rd occurrence on
  - discount recent history as unstable if the user's choices for a
    format have been alternating rather than converging
```

```
  - all output text fields must be written in the user's requested language


OUTPUT REQUIREMENT
  "Return strict JSON only, matching the schema below."
)
```

Code 1 – Structure of the System Prompt

```
"{
  "type": "object",
  "properties": {
    "title": { "type": "string" },
    "adaptive_fields": {
      "type": "object",
      "properties": {
        "short_text": { "type": "string" },
        "long_text": { "type": "string" },
        "image_single_pieces": { "type": "string" },
        "image_assembly": { "type": "string" },
        "video": { "type": "string" }
      },
      "required":      ["short_text",      "long_text",      "image_single_pieces",
"image_assembly", "video"]
    },
    "initial_visibility": {
      "type": "object",
      "properties": {
        "short_text": { "type": "boolean" },
        "long_text": { "type": "boolean" },
        "single_pieces": { "type": "boolean" },
        "assembly": { "type": "boolean" },
        "video": { "type": "boolean" }
      },
      "required": ["short_text", "long_text", "single_pieces", "assembly", "video"]
    },
    "explanation_of_changes": { "type": "string" }
  },
  "required":       ["title",       "adaptive_fields",       "initial_visibility",
"explanation_of_changes"]
```

Code 2 – JSON Schema of the Structured Output Returned by the Adaptation Model

3.1.1.1 Implicit personalisation

The implicit personalization component operates automatically at each step transition, adapting the interface without requiring any direct explicit input from the user. At each step, the model receives three types of information that guide the adaptation decision:

1. **Session interaction history**: a recorded sequence of steps completed during the current session, including the format types accessed by the user. This allows the system to identify real-time behavioural patterns and content preferences.
2. **Aggregated behavioural data from previous sessions**: for each available content format, the system provides information on how frequently that format was accessed by prior users. This enables the model to leverage collective user behaviour, particularly when limited individual data is available.
3. **Step payload**: information about the current assembly step, including its type, title, and the available instruction formats. This helps the model adapt the interface according to the specific requirements of the task.

Based on these inputs, the model autonomously determines which content formats should be initially displayed for the upcoming step, providing a natural-language explanation for each decision. As a result, the interface continuously adapts based on both the user's behaviour and the experience of previous users, without requiring any explicit action from the user.

3.1.1.2 Explicit customization

The explicit customization component allows users to actively shape their training experience by expressing preferences and requirements through LLM input. When starting the adaptive training system, users complete an onboarding form (Figure 3) that collects information in four areas:

1. **Instruction language**: users can specify the preferred language for the training content.
2. **Experience and skill level**: users describe their familiarity with the assembly training activity and with similar technical tasks. This information helps the system adjust the level of detail and complexity of the instruction.
3. **Usability aspects**: users can report readability needs, such as preferred font size or text presentation, enabling the system to adjust the visual interface accordingly.
4. **Additional preferences and requests**: an open-ended field allows users to express any further preferences or requirements, such as preferred content format, sentence length, visual style, text formatting, or colour preferences.

After the form is submitted, the user inputs are processed by the model and used to configure the training interface before the first assembly step is displayed. During the training session, the users' explicit preferences are combined with the real-time adaptations generated through implicit personalisation. This allows the system to deliver a more personalised and user-centred experience that reflects both users stated needs and their observed behaviour.

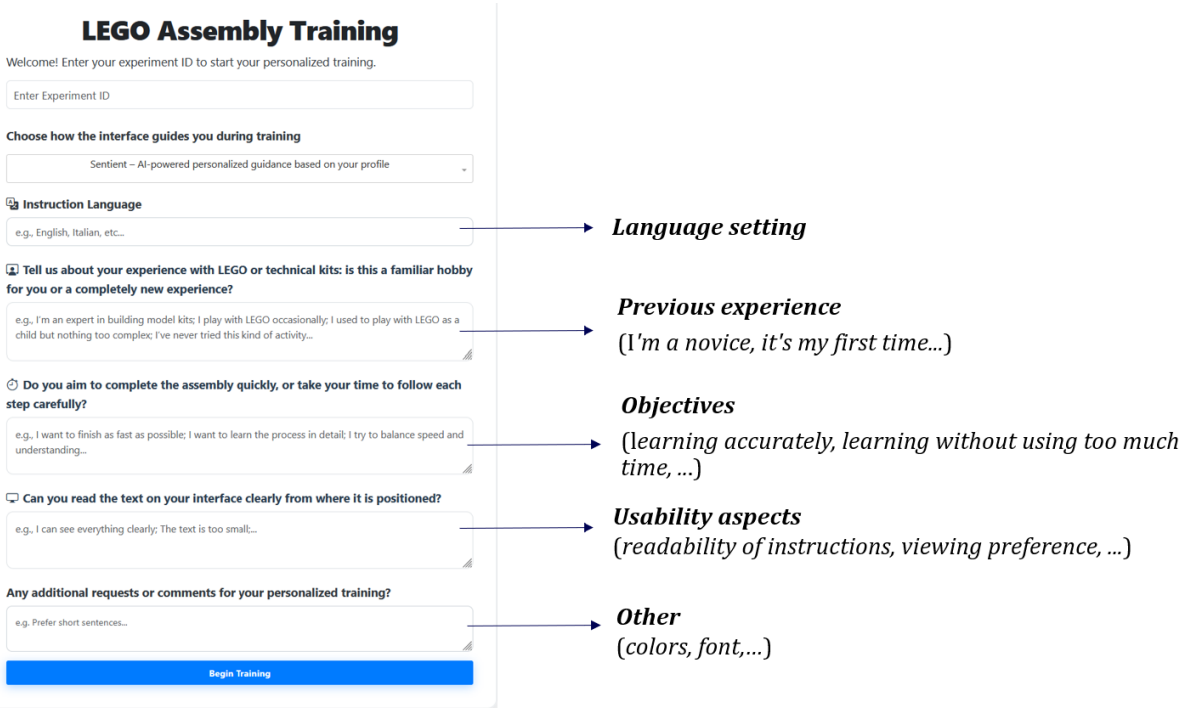


Figure 3 – Onboarding Form for Explicit Customization: Users provide language preferences, prior experience, learning objectives, usability requirements, and additional requests before commencing the training session

3.1.2 Model Evaluation

The adaptive training application presented in this section represents an evolution of the LEGO® Forklift Assembly Training system originally developed and validated in the previous deliverable and further described in detail in [Matteri et al., 2025] [1].

Building upon that foundation, which defined the static and dynamic interface, the interaction protocol, and the rule-based adaptive logic, the system was extended with the GenAI-based adaptation architecture described in the preceding sections, incorporating both the implicit personalization engine and the explicit customization. The result is a fully integrated adaptive training environment that combines the experimental infrastructure of the original system with the new human-centric, GenAI-powered adaptation capabilities developed in the present work.

To illustrate the functioning of the developed adaptive system, a demonstrative prototype of the LEGO® Assembly Training application is accessible via the following link: GenAI demonstration video. The video presents an extract of an adaptive training session, covering the explicit customization onboarding phase and the first three assembly steps. It demonstrates how the interface is configured according to the user's explicit inputs at onboarding and subsequently adapts in real-time based on implicit personalization signals as the session progresses. Figure 4 shows an illustrative screenshot extracted from the video content.

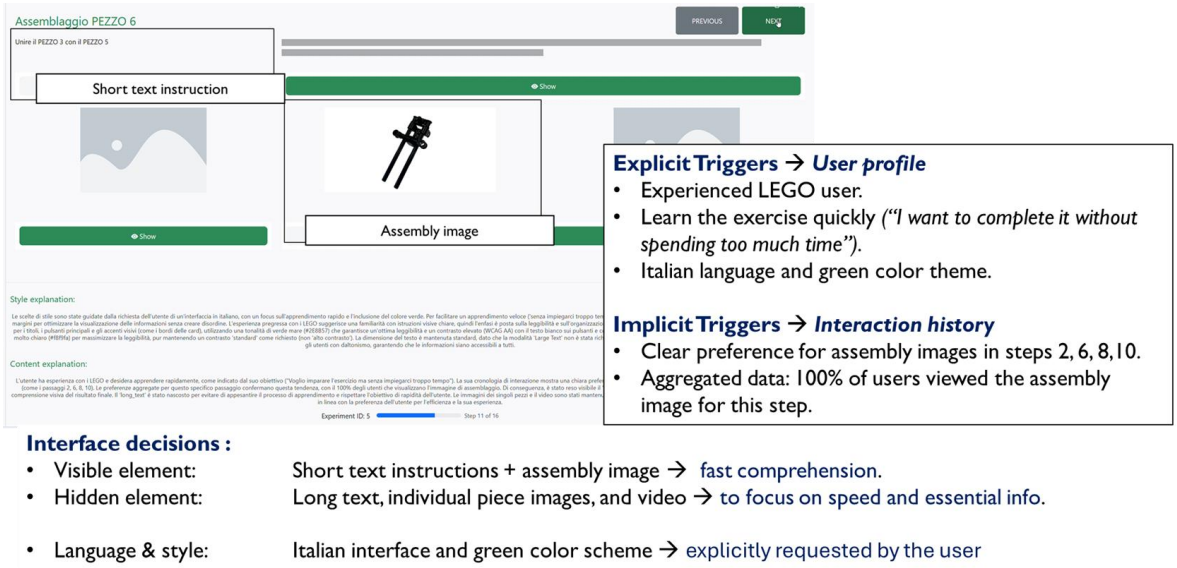


Figure 4 – Screenshot as an Illustrative Example of the Adaptive Model Interface at Step 16 with Notes on Component Details and Explanations

A distinctive feature of the prototype is the integration of an XAI-enhanced interface, which displays a real-time explanation panel at the bottom of the screen at each step. This panel provides a natural-language account of the AI's adaptation decisions, detailing both the Content explanation (which instructional formats were made visible and why) and the Style explanation (visual style choices applied). This transparency mechanism is designed to make the AI's reasoning process legible to the user, supporting informed interaction and fostering trust in the adaptive system.
Figure 5 and Figure 6 present illustrative screenshots of the system in operation, highlighting how explicit user inputs and implicit behavioural data are processed within the adaptive personalization framework.

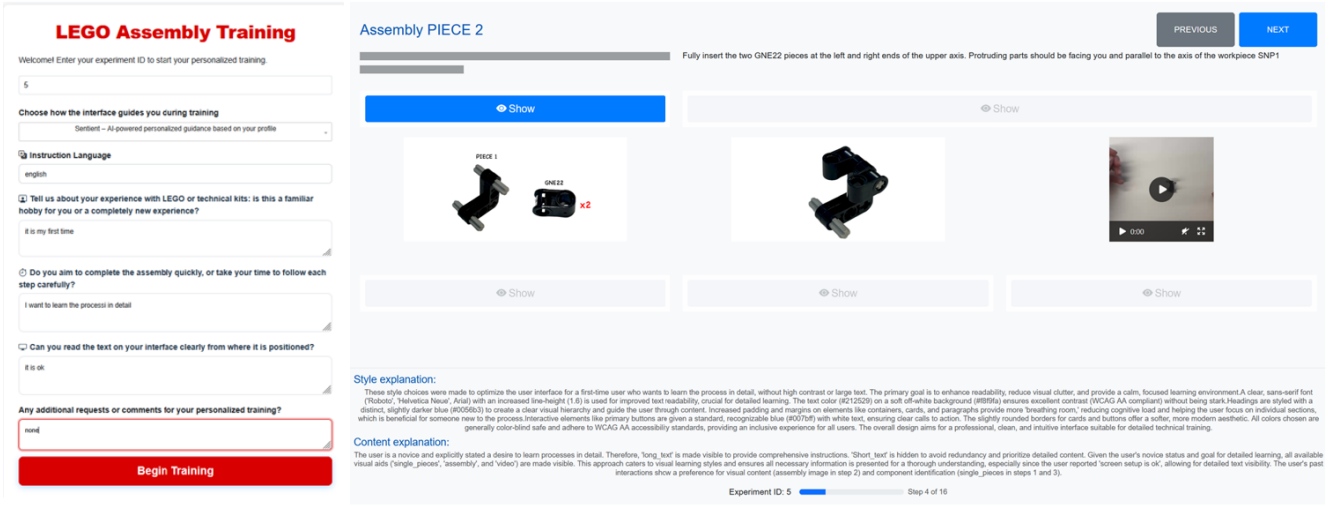


Figure 5 – Illustrative Example of the UI for User ID 5 at Step 4. The screenshot shows the content format selections generated by the system in response to the user's explicit preferences and requests

```
================================================================================
USER 5 - STEP 4
================================================================================
CURRENT USER (5) PREVIOUS INTERACTIONS:
  Step 1: viewed [short_text, single_pieces]
  Step 2: viewed [assembly]
  Step 3: viewed [short_text, single_pieces]

OTHER USERS' AGGREGATED PATTERNS:
  Based on data from 4 previous users:
    Step 1: short_text(100%), single_pieces(50%)
    Step 2: short_text(75%), single_pieces(50%), assembly(75%)
    Step 3: short_text(75%), single_pieces(50%)
    Step 4: short_text(50%), single_pieces(50%), assembly(75%), video(50%)
================================================================================
Aggregated prefs for step 4: {'short_text': 0.5, 'long_text': 0.25, 'single_pieces': 0.5, 'assembly': 0.75, 'video': 0.5}
Gemini adaptation: The user is a novice and explicitly stated a desire to learn processes in detail. Therefore, 'long_text' is made visible to provide comprehensive
instructions. 'Short_text' is hidden to avoid redundancy and prioritize detailed content. Given the user's novice status and goal for detailed learning, all available
visual aids ('single_pieces', 'assembly', and 'video') are made visible. This approach caters to visual learning styles and ensures all necessary information is
presented for a thorough understanding, especially since the user reported 'screen setup is ok', allowing for detailed text visibility. The user's past interactions show
a preference for visual content (assembly image in step 2) and component identification (single_pieces in steps 1 and 3).
After validation: {'short_text': False, 'long_text': True, 'single_pieces': True, 'assembly': True, 'video': True}
================================================================================
```

Figure 6 – Representation of the Implicit User Data Exploited by the System, together with an Explanation of the Adaptive Reasoning Process and the Decisions Adopted by Gemini to Personalize Recommendations and Content Configurations

The two personalization mechanisms underlying this system were assessed through two distinct evaluation approaches, reflecting their different nature and maturity level. The implicit personalization engine, being responsible for the quantifiable, automated prediction of content visibility, was subject to a structured quantitative validation against three reference engines. The explicit customization component, by contrast, was assessed through expert-based verification by the development team, focused on confirming the correct interpretation and integration of user-stated preferences rather than on a structured quantitative comparison.

Adaptive Engine Implementations

To isolate the contribution of within-session behavioural information from the contribution of how the adaptation logic itself is produced, the implicit personalization component was evaluated against three reference engines.

Deterministic Threshold-Based Reference. This engine implements a frequency-threshold mechanism representing standard data-driven practice for adaptive interfaces in industrial training. For each combination of step s and format f, the engine computes the proportion of training users who accessed that format at that step, $\hat{p}(s, f)$. If this proportion meets or exceeds a predefined threshold ($\theta = 0.5$), the format is included in the initial visible set; otherwise, it is hidden:

$$\hat{y} = 1 \; if \;\; \hat{p}(s, f) \geq \theta, 0 \; otherwise$$

This engine depends exclusively on population-level aggregate statistics and does not incorporate any signal from the specific user being served at inference time, meaning it cannot track an individual operator's deviation from the population mean within a session.

Within-Session Threshold Engine. This engine extends the deterministic reference so that its prediction for a given user is informed not only by the population but also by that user's own behaviour earlier in the

same session. For step *s* and format *f*, the engine blends the population proportion $\hat{p}(s, f)$ with the user's own access rate $\hat{p}_{self}(s, f)$, computed over that user's previously completed steps of the same type, through evidence-weighted shrinkage:

$$\hat{p}_{blend}(s, f) = \frac{n \cdot \hat{p}_{self}(s,f) + k \cdot \hat{p}(s,f)}{n+k}, \qquad \hat{y} = 1[\hat{p}_{blend}(s, f) \geq \theta]$$

where *n* is the number of preceding same-type steps the user has completed, and *k* is a pseudo-count expressing the strength of the population prior. An exploration warm-up suppresses the within-session signal for the first three occurrences of each step type, reflecting the assumption that an operator needs to encounter a step type several times before a stable preference can be inferred. This engine uses the same class of information as the LLM-based engine (population statistics together with the user's accumulating within-session history), and therefore serves as the primary, information-matched baseline against which the LLM-based engine is compared.

LLM-Based Adaptive Engine. As described above, this engine receives the same population-level and within-session information as the within-session threshold engine, but expresses the adaptation logic entirely through a natural-language prompt rather than a hand-designed blending rule, requiring no numerical calibration.

In summary, the three engines differ in the information they exploit: the deterministic reference relies solely on population-level statistics; the within-session threshold engine adds the user's own accumulating behaviour through a hand-designed blending rule; and the LLM-based engine uses the same information as the latter, but expresses the adaptation logic through a natural-language prompt rather than a fixed rule.

Validation

The effectiveness of the implicit personalization component was evaluated through a controlled offline study involving 20 participants. Each participant executed the LEGO® forklift assembly training following the 16-step procedure covering three operational categories: component picking, assembly, and quality control. During data collection, participants interacted with a static version of the training interface in which all instructional formats were initially hidden at each step. Users could voluntarily access the available instructional resources by selecting the corresponding interface elements, and every interaction was recorded through timestamped event logs. No adaptive mechanism was active during these sessions, ensuring that the collected interaction patterns represented unbiased user preference.

The recorded logs were subsequently used as ground truth for the offline evaluation of the three adaptive engines described above. For each participant and each assembly step, all three engines generated a binary prediction indicating which instructional formats would have been proactively displayed, compared against the formats actually accessed by the participant. Format-prediction accuracy was adopted as the primary evaluation metric, defined for step *s* and user *u* as the proportion of correctly predicted formats among those actually available at that step:

$$acc(s, u) = \left(\frac{1}{|F_s|}\right) \cdot \sum_{f \in F_s\}} 1[\hat{y}(s, f) = y(s, u, f)]$$

where F_s is the set of instructional formats available at step *s*, y(s, u, f) is the ground-truth access indicator (whether user *u* actually accessed format *f* at step *s*), and ŷ(s, f) is the model's predicted visibility for that format. Formats that are structurally unavailable for a given step (e.g. assembly image on a picking step) are excluded from the calculation, so that the metric reflects genuine adaptive performance rather than

being inflated by trivially correct negative predictions. Per-user accuracy is obtained by averaging acc(s, u) over all 16 steps, and overall accuracy by averaging across all users and steps.

All three engines were evaluated using the same Leave-One-Out Cross-Validation (LOO-CV) protocol: at each iteration, one participant was reserved for testing while the remaining participants were used to estimate step-specific access frequencies (for the deterministic and within-session threshold engines) or to build the behavioural context provided to the LLM (for the LLM-based engine). Because the LLM-based engine's predictions are subject to a non-zero sampling temperature, the full LOO-CV procedure was repeated three times for this engine, and the resulting accuracies were averaged per user.

Results

Under the unified LOO-CV protocol, the within-session threshold engine achieved the highest mean format-prediction accuracy at 81.3% (SD = 7.6%), followed by the LLM-based engine at 79.7% (SD = 5.5%), compared with 74.9% (SD = 11.5%) for the deterministic reference. Both within-session-aware engines therefore improved over the population-only reference, indicating that conditioning on the operator's own in-session behaviour is the primary driver of the observed gains.

A more practically relevant difference emerged in the consistency of predictions across individual users. The deterministic reference's per-user accuracy spanned a range from 57.5% to 96.3% (approximately 39 percentage points), while this spread narrowed to approximately 32 percentage points for the within-session threshold engine and to approximately 22 percentage points for the LLM-based engine, which showed the most uniform performance across participants. This indicates that engines incorporating within-session behavioural information provide more consistent personalisation across heterogeneous users, reducing the risk of poorly matched interface configurations for any individual operator, a property particularly relevant given the multigenerational composition of the manufacturing workforce. Table 1reports overall accuracy for the three engines.

Table 1 – Overall Format-Prediction Accuracy under LOO-CV

Engine	*Mean*	*SD*	*Min*	*Max*
Deterministic (reference)	74.9%	11.5%	57.5%	96.3%
Within-session threshold	81.3%	7.6%	65.9%	97.5%
LLM-based	79.7%	5.5%	67.5%	89.4%

Table 2 reports accuracy disaggregated by step type. Both within-session-aware engines markedly outperform the deterministic reference on picking steps, where only two instructional formats are available, with the LLM-based engine showing the largest margin (+17.3 percentage points). On assembly steps, where all five formats must be predicted jointly, the within-session threshold engine performs best, while the LLM-based engine trails both other engines, reflecting a tendency to over-predict specific formats in richer format spaces. On the single quality-control step, the three engines perform comparably, although the limited number of observations warrants a cautious interpretation.

Table 2 – Prediction Accuracy by Step Type

Step type	*Nr*	*Deterministic*	*Within-session*	*LLM-based*
Picking (2 formats)	140	68.2%	81.8%	85.5%
Assembly (5 formats)	160	82.2%	83.2%	76.5%
Quality control (4 formats)	20	62.5%	62.5%	64.6%

Table 3reports accuracy disaggregated by instructional format. Assembly image is the format on which all three engines perform best and most similarly, consistent with its strongly skewed population-level access

rate, which makes it relatively easy to predict regardless of the adaptation strategy. The LLM-based engine shows its clearest advantage on short text, benefiting from a lower tendency to miss formats that users actually accessed. Conversely, component image is the format on which the LLM-based engine is least reliable, showing a higher tendency to display this format more often than warranted by users' actual behaviour; since this format is available across all three step types, its visibility decision is less constrained by task structure and more dependent on the model's interpretation of limited contextual evidence. The within-session threshold engine, by contrast, performs more consistently across formats, reflecting the corrective effect of blending population statistics with the user's own access history.

Table 3 – Performance by Institutional Format

Format	*Deterministic*	*Within-session*	*LLM-based*
Short text	69.7%	79.7%	78.9%
Long text	82.2%	85.6%	79.8%
Component image	75.0%	79.4%	70.3%
Assembly image	93.1%	93.1%	94.0%
Video	77.2%	73.9%	73.1%

Overall, these results support the viability of LLM-based adaptive engines as a calibration-free alternative for adaptive industrial training systems: the LLM-based approach matches the accuracy of an explicitly hand-tuned, within-session-aware rule while offering more consistent personalisation across operators than a population-only baseline, a property of particular value in deployment scenarios characterised by heterogeneous operator populations or limited historical data.

3.1.3 Summary

While the original system relied on a dynamic rule-based engine, driven by historical user behaviour and real-time click-stream tracking, to reactively adjust the visibility of instructional content, the new GenAI-powered architecture introduces a fundamentally different approach to interface adaptation, expanding both the scope of what is adapted and the intelligence with which adaptation decisions are made.

The most significant shift lies in the nature of the input streams. The rule-based system operated exclusively on population-level behavioural data, applying pre-set thresholds to determine content visibility through semi-automated triggers. The GenAI-powered system, by contrast, adopts an input model that combines implicit behavioural tracking, drawing on both within-session interaction history and aggregated prior-user preferences, with explicit user input collected at onboarding through natural language prompts. This dual-stream architecture is what enables the system to move beyond generic one-size-fits-all personalisation toward a genuinely hyper-personalised experience.

This expanded input model also enables a broader range of interface adaptations. Where the previous system addressed only content adaptation, controlling the visibility of images, video, and text, the GenAI-powered system operates across the full interface stack, encompassing both content selection and visualisation aspects such as language, text formatting, and dynamic UI aesthetics. Furthermore, the response type has evolved from reactive to predictive, with the model anticipating user needs based on accumulated behavioural evidence and explicit preferences before each step is presented.

From an architectural standpoint, the transition from a semi-automated, rule-based engine to an autonomous AI agent also changes the learning dynamics of the system. The rule-based approach required periodic manual updates to incorporate new behavioural data, whereas the GenAI-powered system supports continuous learning through real-time model refinement, accumulating collective user knowledge with each new session and progressively improving its adaptation decisions without developer intervention. These differences are summarised in Table 4.

Table 4 – Benchmark: Dynamic Rule-Based VS GenAI-Powered

	Dynamic rule based	*GenAI-powered system*
Input	Historical user behaviour + real-time click-stream tracking	Explicit LLM prompts + implicit behavioural tracking
Interface elements	Content (visibility of images, video, text)	Content and visualisation (language, dynamic UI aesthetics)
Response type	Reactive (triggered)	Predictive (anticipates needs)
System training	Periodic manual updates	Continuous learning (real-time model refinement)
Automation level	Semi-automated (pre-set rules)	Autonomous agent
User experience	Personalised	Hyper-personalisation

3.1.4 Main Outcomes

The tangible outputs from Task 3.5 serve as foundational components for subsequent activities across the XR5.0 project:

- Refined AUI Design Framework: the updated methodology, specifically the clarified "Adaptation Mechanism" and streamlined design protocol, provides a standardized blueprint for researchers and designers to replicate or adapt these interfaces.
- GenAI-Powered Prototype: the developed training application serves as a demonstrator for integrating XAI (Explainable AI) transparency into industrial interfaces, setting a standard for fostering trust between human operators and AI agents.
- Validation Protocol: the offline study methodology, utilizing timestamped interaction logs as a ground truth, establishes a new benchmark for evaluating personalization engines that can be applied to future XR content adaptation tasks.

4 CLAWDITE AND THE ADAPTATION ENGINE FOR THE TRAINING PLATFORM

This chapter describes the Adaptation Engine, the functional module through which the adaptive-training logic presented in the previous chapters is delivered to XR clients and integrated with the XR5.0 Training Platform and the Clawdite HDT infrastructure.

Whereas the section *The Framework for AUI Design* addressed the design methodology and the section *Algorithms for Adaptive Training* addressed the GenAI-powered adaptation logic, this chapter focuses on the engine that operationalises them: its API contract, its data model, and the way training content authored in the Training Platform and contextual data maintained in Clawdite flow through it to produce per-element and per-path presentation decisions. The module was introduced in Deliverable D3.2 as the *UI Adaptation Engine* proof of concept and is jointly developed by SUPSI and IML. The current release, *adaptation-engine v0.2.0*, is deployed on the shared XR5.0 Clawdite instance (https://xr50.clawdite.spslab.ch/adaptation-engine/) and documented through an interactive OpenAPI specification (https://xr50.clawdite.spslab.ch/docs/).

Within the XR5.0 architecture the engine occupies the interface-adaptation layer that sits on top of two data backbones as depicted in Figure 7. The Training Platform (IML), comprising the Training Programs Authoring Tool (TPAT) and the XR Training Plugin (XRTP), owns the training content (i.e., programs, workflows, checklists, images, videos, quizzes and 3D models) and exposes it through its content API. Clawdite owns the HDT, including the worker profile and the multimodal human-state indicators (e.g., the real-time stress estimate) described in D3.2. The Adaptation Engine consumes content from the former and context derived from the latter and returns presentation decisions that the XR clients apply to their interfaces. Consistently with the original proof of concept, the engine is framework-independent: it communicates exclusively through a REST API exchanging JSON, so the same service drives both web front-ends and the Unity-based headset UIs used across the pilots, and can be integrated with any system able to process real-time JSON-based UI configuration updates.

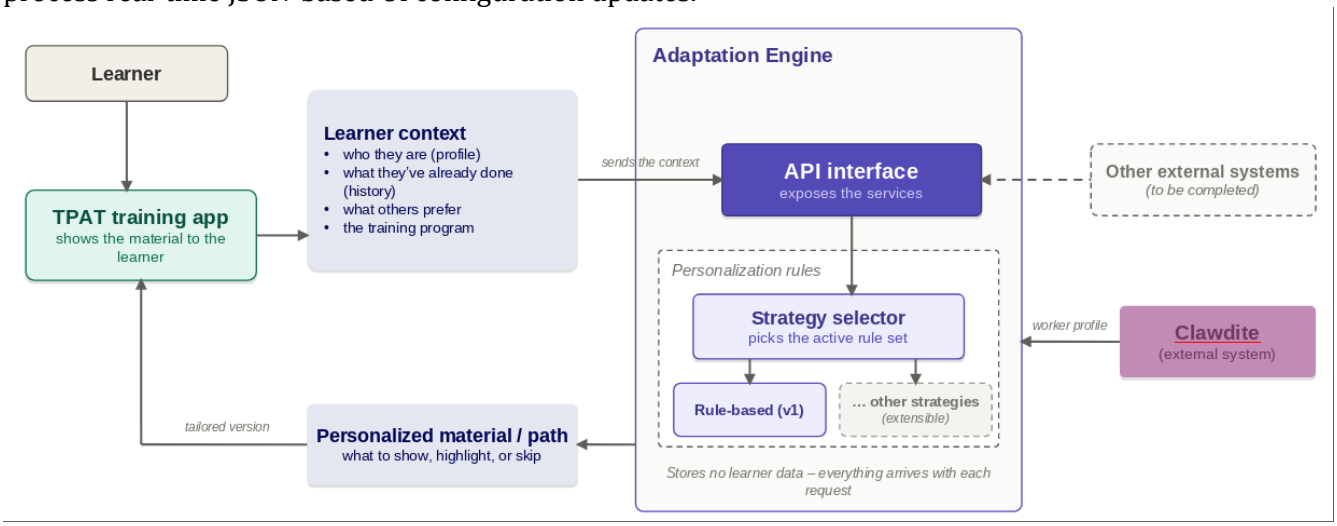


Figure 7 – Adaptation Engine within the XR5.0 Architecture

4.1 From the UI Adaptation Engine Proof of Concept to a Content-Aware Service

In D3.2 the module was presented as a proof of concept providing adaptive and remotely configurable user-interface behaviour, exchanging configuration data in JSON and validating every incoming parameter against a predefined *adaptation space* to preserve UI consistency and prevent invalid states. That first version reasoned at the level of low-level interface primitives, through menu layouts, button properties and spacing rules.

Driven by the requirements of the XR5.0 pilots, the scope has since broadened from widget geometry to the training content itself: the engine now decides, for a single unit of material and for an entire learning path, what a given worker should see, expand, annotate, collapse or skip by default. The guiding principles of the prototype are preserved and given a stronger formal basis, the framework-independent JSON/REST

interface is realised through two typed endpoints, and the bounded adaptation space is enforced by a deterministic validation filter that guarantees the client never receives an invalid or unsafe instruction.

4.2 Design Rationale

Five design decisions define the service contract and follow from the role the engine plays between the Training Platform and the XR clients.

- **Stateless operation, context per call:** the engine retains no per-user session, progress record or interaction log. Every input a decision depends on (past interactions, aggregated peer behaviour and the surrounding learning path) is supplied in the request as a first-class context object. This makes the service horizontally scalable, side-effect free and easy to test, and leaves the client (which already owns the content and the user session) in control.
- **Strong typing over a closed set of material types:** there are exactly six material shapes (checklist, workflow, image, video, quiz and unity), each modelled as a typed object. A material is sent as received from the content API, validated against its schema, and answered with an adaptation of the matching type.
- **Uniform handling of related references:** every material, and every element within it, may carry references to other materials. The engine treats the visibility and inlining of these references as an adaptable property wherever it appears, through a single `RelatedAdaptation` structure applied at every level.
- **Pluggable strategy with a deterministic safety net:** personalisation logic is encapsulated in a named strategy; new strategies, including the GenAI-powered one described earlier in this deliverable, can be added server-side without changing the API. Independently of the selected strategy, every output passes through a deterministic validation filter that cannot be bypassed.
- **Separation of subject and context:** each request distinguishes the subject of the adaptation (a material, or a program for path orchestration) from the context against which it is adapted. The context object is identical across both endpoints; only the subject differs.

4.3 Personalisation Operations and the Context Model

The service exposes exactly two personalisation operations. **Per-material adaptation** takes one material plus context and returns per-element visibility, expansion and annotation decisions, answering "given this worker, how should this material be presented?". **Path orchestration** takes a program plus context and returns skip, collapse and reorder decisions over the learning path, answering "which materials in this program should this worker work through?". Content storage and retrieval, rendering, and the persistence of user state are explicitly outside the service.

4.3.1 The Context Object

Both endpoints accept the same context bundle, `{ profile, history, peer_preferences, program }`. Every field is optional and defaults to empty, so a cold-start client may legitimately send an empty context.

- **profile:** the worker's `language`, `self_declared_level` (an integer competency rank driving gating and the novice/expert rules) and `device_type` (`desktop`, `tablet` or `ar_vr`), together with free-text fields (prior experience, training objective, usability and visual-comfort needs, other requests).
- **history**: the cumulative list of past interaction events (`material_id`, `sub_element_id`, `action`, timestamp), enabling history-based personalisation in which elements the worker repeatedly opens are favoured.
- **peer_preferences**: aggregated open-frequencies across the workforce, indexed as `peer_preferences[material_id][element_id]` in the range [0, 1]; tabulated by an analytics job, not by the engine.

- **program**: on the path endpoint, the program being orchestrated (required); on the material endpoint, the surrounding learning path (optional), giving the strategy visibility into sibling materials.

This model is the natural meeting point of the two personalisation streams described in the section *Algorithms for Adaptive Training*: the explicit user inputs collected at onboarding map onto the `profile`, while the implicit behavioural signals map onto `history` and `peer_preferences`. It also provides the insertion point for the human-state indicators maintained in Clawdite: the worker profile and competency rank are already part of the HDT, and the confidence-aware stress and cognitive-load estimates produced by the Stress Index Calculator (D3.2) can be surfaced through the same context object, allowing future strategies to relax or intensify assistance according to the worker's current state.

4.3.2 Competency Gating of the Learning Path

Path orchestration is governed by competency gates carried by each item and compared against the worker's `self_declared_level`:

- `required_upto_level_rank = R`: mandatory for workers whose level is ≤ R; for workers above R it becomes a candidate to skip or collapse.
- `min_level_rank` / `max_level_rank`: a hard range; a worker outside it has the material skipped unconditionally.
- No gates: the material is always shown.

4.4 Data Model: Materials and Adaptation Decisions

Every material carries the common fields `id`, `name`, `description`, `unique_id`, `created_at`, `updated_at`, `related[]` and the `type` discriminator. Related references are shaped `{ id, name, description }` and appear at both material and element level. Unknown extra fields are ignored, so additions to the upstream content schema do not break the integration. The six material types and their adaptable elements are listed in Table 5.

Table 5 – The Six Supported Material Types and the Elements the Engine can Adapt

Type	*Adaptable Elements*
`checklist`	config.entries[] — each with text, description and related references
`workflow`	config.steps[] — each with title, markdown content (may contain CAUTION blocks and checkboxes) and related references
`image`	config.annotations[] — each with text, position and related references
`video`	config.timestamps[] — chapters and annotations, each with related references
`quiz`	config.questions[] — each with answers; both questions and answers carry related references
`unity`	recursive parts in the model tree — each with visibility/annotation flags and an explode amount

The output is typed symmetrically: the adaptation type always matches the material type. Each decision is a map of overrides keyed by the element id taken from the submitted material, for example entry visibility for checklists, step expansion and CAUTION-block visibility for workflows, or per-part visibility and explode amount for Unity models. Related references are handled everywhere through the same `RelatedAdaptation` structure, whose visibility controls whether a reference is shown and whose expansion controls whether it is rendered inline rather than collapsed. Crucially, these maps express

defaults, not constraints: an element id absent from a map keeps the client's own default, and every element remains user-toggleable, with the resulting toggles recorded in the interaction history so that subsequent calls adapt accordingly.
A companion object, `enabled_interactions`, declares which interactions the current screen can technically render and acts as the upper bound on the output: the engine ANDs every visibility decision against it, so no strategy can expose an interaction the screen cannot support. The client derives these flags from the payload and its own rendering capabilities, for instance, enabling a related-references flag only when the corresponding list is non-empty, permitting autoplay only where device policy allows, and gating the Unity flags on the capabilities of the 3D viewer in use.

4.5 Adaptation Strategies and the Deterministic Safety Net

Personalisation logic is selected per request through a strategy name. The engine ships a deterministic, transparent default, `rule_based_v1`, whose output is predictable and straightforward to verify during quality assurance. It evaluates the worker's declared level (novice ≤ 2, expert ≥ 4, intermediate in between), the device type, the interaction history (recent opens weighted higher through an exponential decay) and the peer preferences, and accompanies every response with an explanation and the list of applied rules. Its behaviour per material type is summarised as follows: for checklists, experts have descriptions hidden and repeated entries grouped while novices have related references expanded inline; for workflows, CAUTION blocks are forced visible, experts see steps collapsed and novices expanded, and on AR/VR related image and 3D references are expanded; for images, experts see only the first annotations; for videos, chapters are always visible and annotations shown for novices; for quizzes, help text is hidden for experts; and for Unity models, non-experts have every part annotated and the exploder opened wider.
The **GenAI-powered adaptation logic** described in the section *Algorithms for Adaptive Training* is integrated as a second, pluggable strategy that shares the same request/response contract: the client simply selects a different strategy name and supplies the matching parameters. This is what allows the LLM-based engine, already under test on the LEGO assembly case, to be served through the same infrastructure as the deterministic baseline, and it is the mechanism by which the two approaches benchmarked earlier in this deliverable are deployed interchangeably. The tunable parameters of the deterministic strategy are summarised in Table 6.

Table 6 – Tunable Parameters of the Deterministic rule_based_v1 Strategy

Parameter	*Default*	*Meaning*
`history_bias_weight`	0.5	Weight of the worker's own open-history.
`history_decay_half_life_steps`	5	Recency half-life for history scoring.
`peer_bias_weight`	0.3	Weight of peer preferences.
`visibility_threshold`	0.5	Score above which an element defaults to visible.
`novice_level_threshold`	2	Level ≤ this value applies "novice" rules.
`expert_level_threshold`	4	Level ≥ this value applies "expert" rules.
`level_rank_strict`	true	true skips gated path items; false collapses them.

Every response is post-processed by a deterministic filter that runs after the strategy: the strategy proposes, the filter enforces. Regardless of which strategy is selected, the filter drops decisions that reference non-existent elements, ANDs every visibility flag with the corresponding `enabled_interactions` flag, forces workflow CAUTION visibility off when caution blocks are disabled on the screen, drops Unity overrides for absent parts, and prunes related-reference decisions to ids that exist. For path orchestration it guarantees

that mandatory materials cannot be skipped (any such proposal is reverted to visible), that out-of-range materials are force-skipped, and that overrides for unknown materials are dropped. This filter is the formal realisation of the bounded "adaptation space" principle introduced in D3.2: the client can never be instructed to show a non-existent or disabled element, nor to skip a material the worker is required to complete. Safety is therefore a property of the service rather than of any individual strategy, which is precisely what makes it safe to add experimental or LLM-backed strategies.

4.6 Service Interface and Division of Responsibilities

The service exposes a small REST surface. All adaptation endpoints require a JSON Web Token (JWT) obtained from the authentication endpoint.

Table 7 – REST Interface of the Adaption Engine

Endpoint	*Purpose*
`GET /auth`	HTTP Basic → JWT exchange (one-hour token expiry).
`GET /adaptation/strategies`	List the registered strategies and their parameter JSON schemas.
`POST /adaptation/material`	Adapt one material.
`POST /adaptation/path`	Orchestrate a program's learning path.

Because the service is stateless, responsibilities are cleanly divided between the engine and the client, which is what allows Clawdite, the Training Platform and the engine to evolve independently:

- **Client** (Training Platform front-end / XR client): fetching programs and materials from the content API, deriving the enabled-interactions flags, maintaining the interaction history and user profile (with HDT data from Clawdite), aggregating peer preferences, and rendering content and applying the returned decisions.
- **Adaptation Engine:** deciding what to show, expand or skip, and guaranteeing through the validation filter that the decisions are valid and safe.

4.7 Integration with Clawdite and the Training Platform

Interface adaptation in XR5.0 is the result of a coordinated flow across three components. The Training Platform stores programs and materials and exposes them through its content API; Clawdite provides the HDT context (worker profile and competency rank and, prospectively, the human-state estimates from the other Functional Modules); and the Adaptation Engine combines a material or a whole program with this context and returns presentation decisions, which the XR client (typically a Unity headset application) applies over its baseline rendering. The resulting client-side call lifecycle should be as follows:

- At application start, the strategies endpoint is queried once to confirm the available strategies and read their parameter schemas.
- On authentication, the client obtains and caches a JWT, refreshing it on expiry.
- When the worker opens a program, the client forwards it to the path endpoint and applies the skip/collapse decisions to the learning-path UI.
- When the worker enters a material screen, the client derives the enabled-interactions flags, calls the material endpoint and applies the adaptation, keeping all elements user-toggleable.
- Opening a related material triggers a further material call; element toggles are appended to the local interaction history without a service call, so later adaptations reflect the worker's behaviour.

Because the contract is JSON over REST and framework-independent, the same engine serves the Training Platform web front-end and the pilot headset UIs without modification, fulfilling the integration intent

recorded for this module in D3.2: namely, integration into the XR5.0 Training Platform and pilot-specific applications to support dynamic adjustment of menu layouts, button properties, spacing rules and other UI components.

4.8 Future Work and Integrations

With this deliverable, the development activities within WP3 are concluded: the Adaptation Engine is delivered operational under the rule-based strategy and integrated with the Training Platform at interface level. Its consolidation and end-to-end validation will be carried out beyond WP3 as a joint effort of WP5 and WP6 and will be described within D6.4, along the following lines:

- **LLM-backed strategy**: the strategy has been designed so that it can be generalised, rewriting and translating content and populating the currently-empty summary fields, without any change to the request/response contract; clients are therefore advised to treat the strategy name as a configuration value and to tolerate currently-empty fields being populated in the future.
- **Context enrichment from Clawdite**: the context object will be progressively connected to the multimodal human-state indicators maintained in the HDT, allowing adaptations to respond to estimated stress and cognitive load in addition to declared competency, history and peer behaviour.
- **Pilot validation**: the strategy parameters will be calibrated with pilot data and the module evaluated under real operational conditions within WP6, providing quantitative and qualitative evidence of its impact on worker support and XR-based interaction.

5 INTEGRATION IN XR TRAINING PLATFORM AND DIGITAL TWIN

The integration of adaptive XR content within the XR5.0 ecosystem requires the seamless interaction of multiple system components, including the XR5.0 Training Platform, the Adaptation Engine, and the Clawdite platform implementing the HDT. This section describes how the personalisation mechanisms presented in the previous chapters are operationalised and deployed within XR environments, enabling real-time, human-centric adaptation of training experiences.

The work carried out in this section contributes to bridging the gap between adaptive algorithms and their practical application in industrial XR scenarios. In particular, it demonstrates how behavioural data, user preferences, and contextual information derived from the HDT are used to dynamically tailor XR content during training execution.

An overview of the integration is shown in Figure 8.

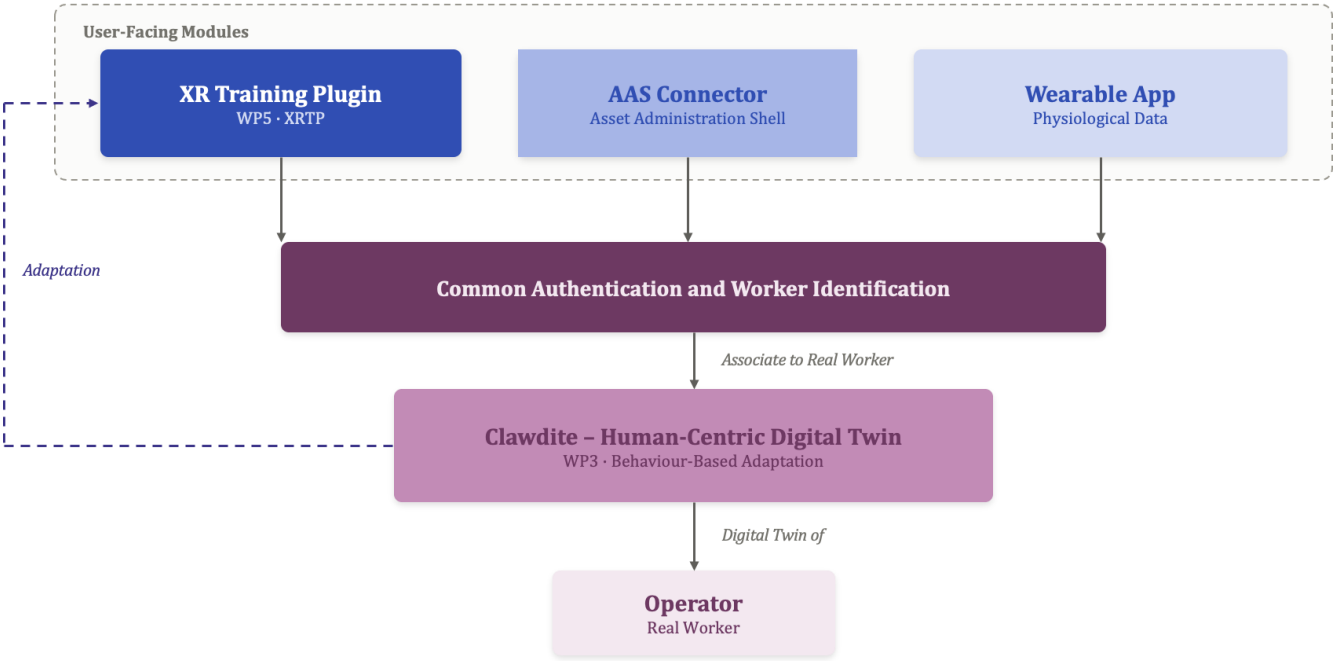


Figure 8 – XR Platform and HDT Integration Architecture

User-facing modules, including the XR Training Plugin (XRTP), wearable applications, and auxiliary services, interact through a unified authentication mechanism to associate each training session with a specific worker entity within the HDT. These modules exchange data with the Clawdite platform, enabling continuous updates of the worker model and supporting real-time adaptation of XR content.

The integration is structured around three main functional aspects:

- **Mental workload and stress monitoring** (Section 5.1): subjective and physiological data collected during XR sessions are integrated into the HDT to enable real-time assessment of operator state.
- **Adaptive XR content personalisation** (Section 5.2): training content is dynamically adapted based on the operator's HDT profile and observed behaviour.
- **Authentication and user identification** (Section 5.3): a unified mechanism ensures consistent identification of users across all XR5.0 components and reliable linkage to the HDT.

5.1 Mental Workload and Stress Questionnaire Integration in VR Environment

To support human-centric adaptation, the XR5.0 Training Platform integrates mechanisms for monitoring mental workload and psychological stress directly within XR environments. These mechanisms are based on the methodologies defined in Deliverable D3.2.
Within the Pilot 4 VR application, questionnaires are presented at predefined stages of the training workflow: prior to task execution, at intermediate points, and upon task completion. The questionnaires are displayed through spatial user interface panels embedded within the XR environment, allowing operators to provide input without interrupting their activity (Figure 9). The assessment includes two self-report scales that measure the operator's subjective stress experience while performing an aircraft maintenance procedure in a virtual reality environment. The Mental Workload scale measures the cognitive demands imposed by the task. The Psychological Stress scale measures the operator's perceived stress level at a given moment. Currently, this module is integrated exclusively within the Pilot 4 application; its integration into the XRTP is planned as a generic solution applicable across all pilots.

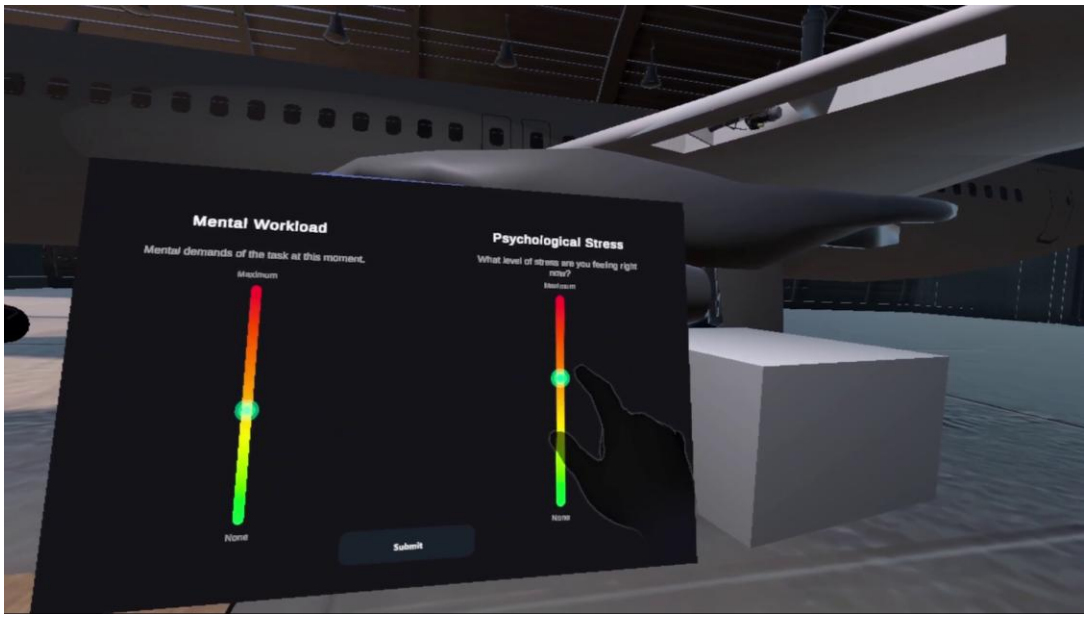


Figure 9 – Mental Workload and Psychological Stress Evaluation Panel in the Pilot 4 VR Application

Upon submission, the collected data is transmitted in real time to the Clawdite platform, which implements the HDT, through an MQTT-based communication channel. This enables continuous acquisition and storage of subjective indicators reflecting the operator's cognitive and emotional state.
The questionnaire data is combined with physiological measurements acquired through wearable devices and processed by the Stress Index Calculator module within the HDT. This integration enables the estimation of the operator's stress level at specific moments during task execution, providing essential input for adaptive XR content personalisation (Figure 10).

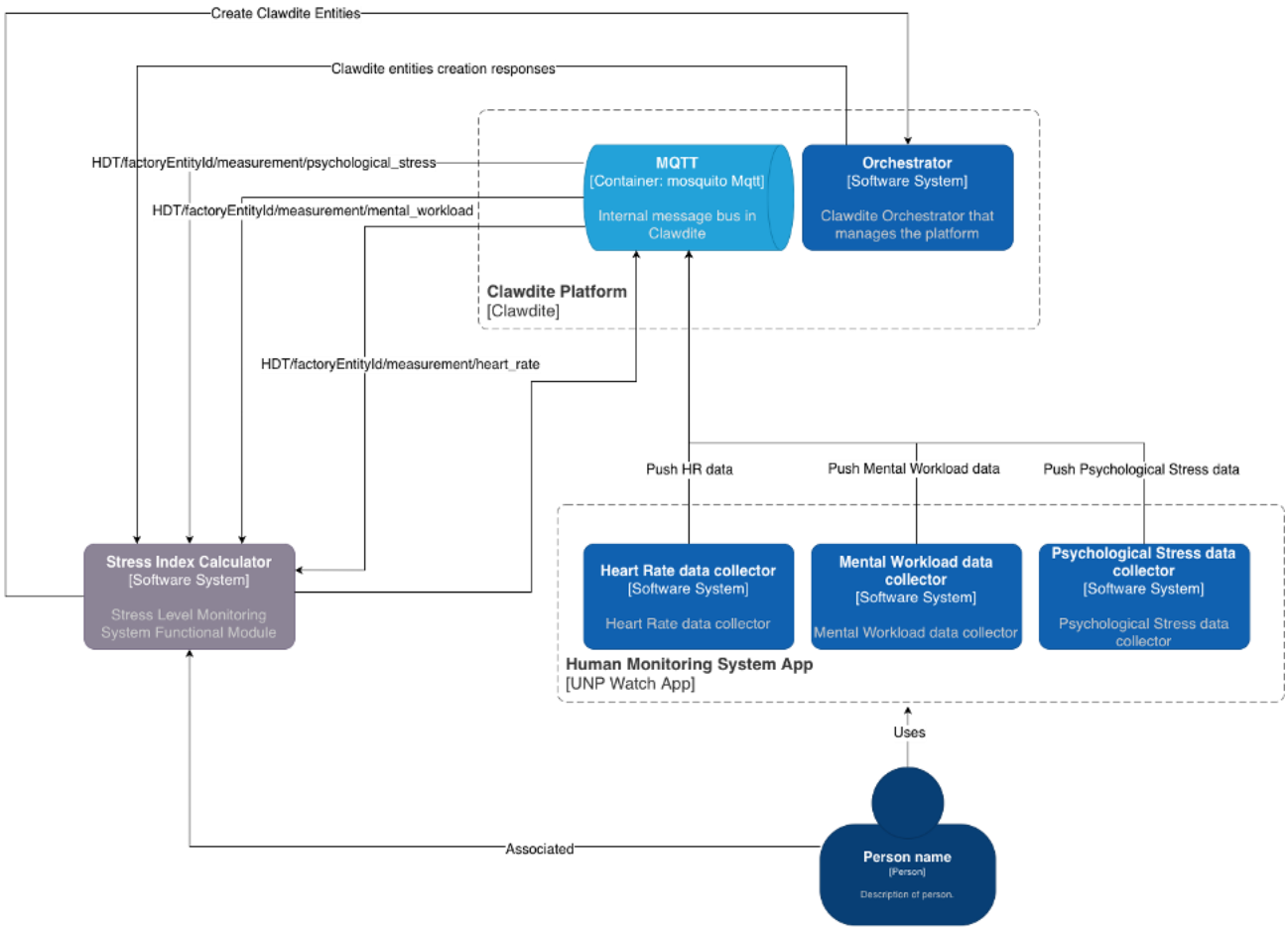


Figure 10 – Stress Index Calculator within the Clawdite Human-Centric Digital Twin: High-Level Functional View

5.2 Adaptive Content Personalisation in the XR Training Platform

The XR5.0 Training Platform enables two complementary levels of XR content personalisation: (i) static customisation and (ii) dynamic adaptation.

Static customisation is based on predefined user attributes, such as experience level, and determines the structure and availability of training content before execution.

Dynamic adaptation operates in real time during training execution. It relies on interaction data collected by the XRTP and contextual information derived from the operator's HDT. This information is transmitted to the Adaptation Engine, which generates personalised recommendations regarding the presentation of training materials (Figure 11).

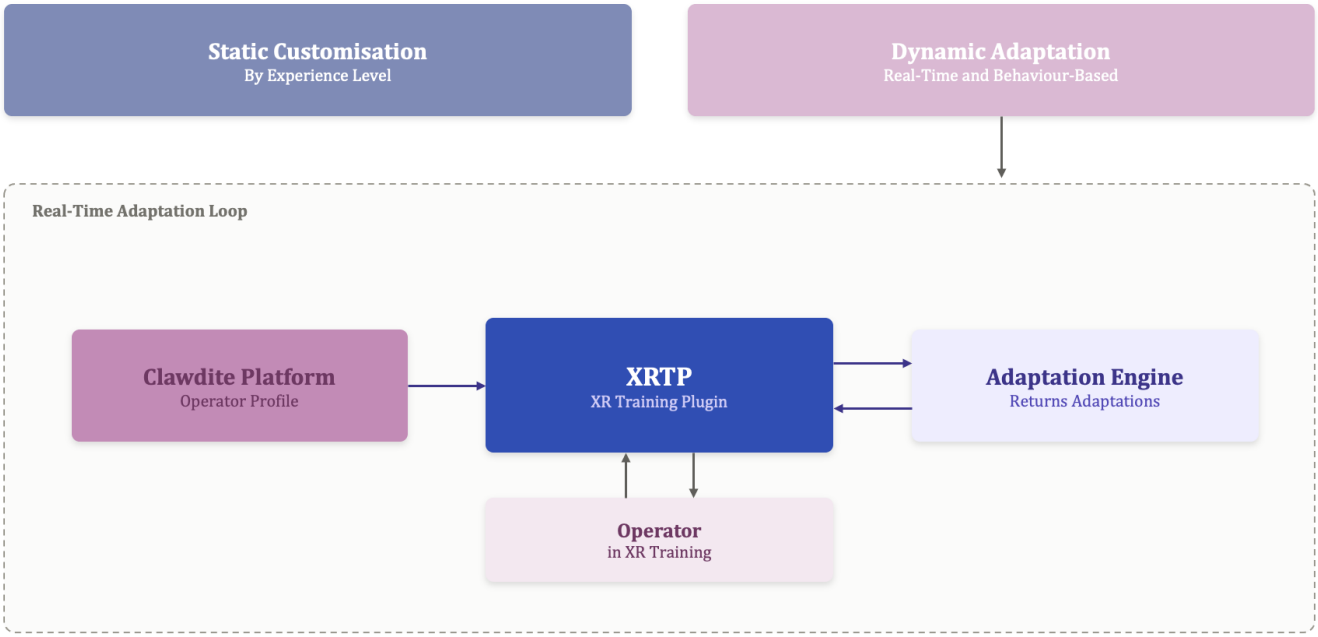


Figure 11 – Static Customisation and Dynamic Adaptation Loop in the XR5.0 Training Platform, driven by HDT Data and User Behaviour

The XRTP applies these recommendations by dynamically modifying the user interface. This includes adjusting the visibility and prioritisation of training elements such as images, checklists, and instructional videos. The objective is to provide the appropriate level of support at each stage of the task while avoiding cognitive overload.

5.3 Authentication and User Identification Solution

This subsection defines the solution adopted in XR5.0 for authenticating users and identifying the corresponding worker entity across the platform. The same mechanism is shared by all operator-facing components, including the XR Training Plugin (XRTP), the Asset Administration Shell (AAS) connector, and wearable applications, providing a unified approach to user authentication and session association. In particular, the solution ensures that each XR session is consistently linked to a specific worker entity within the HDT implemented in the Clawdite platform.

The proposed solution addresses two fundamental aspects:

- **User authentication**, i.e. how operators access the system;
- **Identity resolution**, i.e. how an authenticated user is mapped to a worker entity within the HDT, identified by a unique worker identifier (**workerID**) and associated organisation.

The subsequent use of the workerID by downstream components, such as the transmission of interaction and physiological data into the HDT, lies outside the scope of this subsection. The authentication workflow described here concludes once the client application has obtained a verified workerID and organisational context.

User authentication is provided by Hololight Hub, which therefore acts as the platform's Identity Provider (IdP). Hololight Hub is responsible for verifying user credentials and issuing authentication tokens. In contrast, worker identity information, including the workerID and organisational structure, is maintained within the Clawdite platform, which implements the HDT.

Since Hololight Hub and Clawdite operate as independent systems with no native knowledge of each other, a dedicated mechanism is required to establish a reliable link between an authenticated user and the corresponding worker entity in the HDT. The allocation of responsibilities between the two platforms is summarised in Table 8.

Table 8 – Allocation of Identity Responsibilities between Hololight Hub and Clawdite

Concern	*Owned by*
User identity and login (Authentication)	Hololight Hub
Worker model, organization, workerID	Clawdite (HDT)

Within the Clawdite platform, each worker is represented as an entity identified by a universally unique identifier (UUID), referred to as the workerID, and grouped under an organisation. In addition to core attributes, each worker entity may include a set of configurable characteristics, which are named attribute-value pairs used to store supplementary information about the worker. These characteristics include, among others, an external identity attribute that enables linking the worker entity to an authentication identity issued by Hololight Hub.
Hololight Hub governs user authentication and access control through mechanisms such as Single Sign-On (SSO) and Role-Based Access Control (RBAC). It is designed as an access gateway, while system integration logic, including the mapping between user identity and worker entities, is delegated to external components.
The proposed identity solution relies on a number of architectural assumptions regarding both platforms, which must be validated against the live system instances prior to deployment:

- **Hololight Hub login interface.** A login interface is available that accepts user credentials or executes a SSO flow, returning a session token (assumed to be a JWT or equivalent bearer token). The exact authentication endpoints, token structure, claims, and validity period are defined by the Hololight Hub implementation used in XR5.0.
- **Stable identity claim.** The authentication token (or a related user information endpoint) exposes a stable user identifier, such as an email address, SSO subject identifier, or Hub-specific user ID, which can be used as a consistent mapping key between the authentication system and the HDT.
- **Worker lookup by characteristic.** The Clawdite Orchestrator API supports the retrieval of worker entities based on characteristic values. This capability is essential for implementing the identity-mapping mechanism, as it allows matching the external identity attribute stored in the HDT with the identity claim extracted from the authentication token.
- **Server-side API credential.** Access to the Clawdite Orchestrator API is performed using a secure server-side API key, which is never exposed to client applications. This ensures that sensitive credentials remain within a trusted backend environment.

These premises define the foundation for the identity resolution mechanism described in the following subsections, ensuring a secure, scalable, and interoperable approach to linking authenticated users with their corresponding worker entities in the HDT.

5.3.1 Overview: the Identity Bridge

User authentication is performed via Hololight Hub, which acts as the IdP. However, worker identity, including the workerID and organisational affiliation, is managed within the Clawdite platform (HDT).
To bridge these systems, an intermediate component is introduced: the Identity Bridge. This server-side service is responsible for linking authenticated users to their corresponding worker entities within the HDT. The Identity Bridge validates authentication tokens issued by Hololight Hub, extracts a stable user identifier, and retrieves the associated worker information from the Clawdite platform.

5.3.2 Architecture and Trust Boundary

The architecture of the Identity Bridge introduces a clear trust boundary between client applications and backend services.
Client-side applications interact only with:

- Hololight Hub (for authentication);
- the Identity Bridge (for identity resolution).
- The Identity Bridge communicates with the Clawdite platform using secure server-side credentials, ensuring that sensitive information, such as API keys, remains protected within the trusted backend

environment.

Figure 12illustrates the architecture and trust boundary of the Identity Bridge solution.

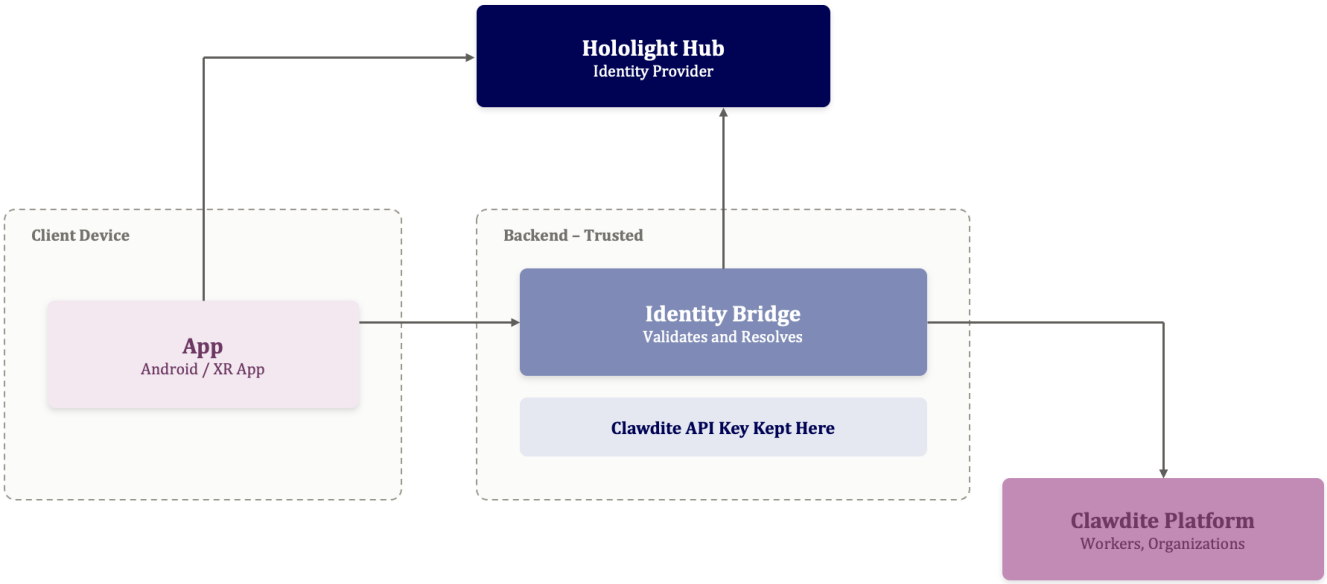


Figure 12 – Identity Bridge Architecture and Trust Boundary between Client Applications and the HDT Backend

5.3.3 Authentication and Resolution Flow

The authentication and identity resolution process follows a structured sequence:

- The client application authenticates the user via Hololight Hub.
- A session token is returned and stored by the client.
- The client sends the token to the Identity Bridge.
- The Identity Bridge validates the token and extracts a stable identity claim.
- The Identity Bridge queries the Clawdite platform (HDT) to locate the corresponding worker entity.
- The workerID and organisational data are returned to the client.
- The client uses this information for downstream processes, including XR content adaptation.

Figure 13 presents the sequence diagram for this process.

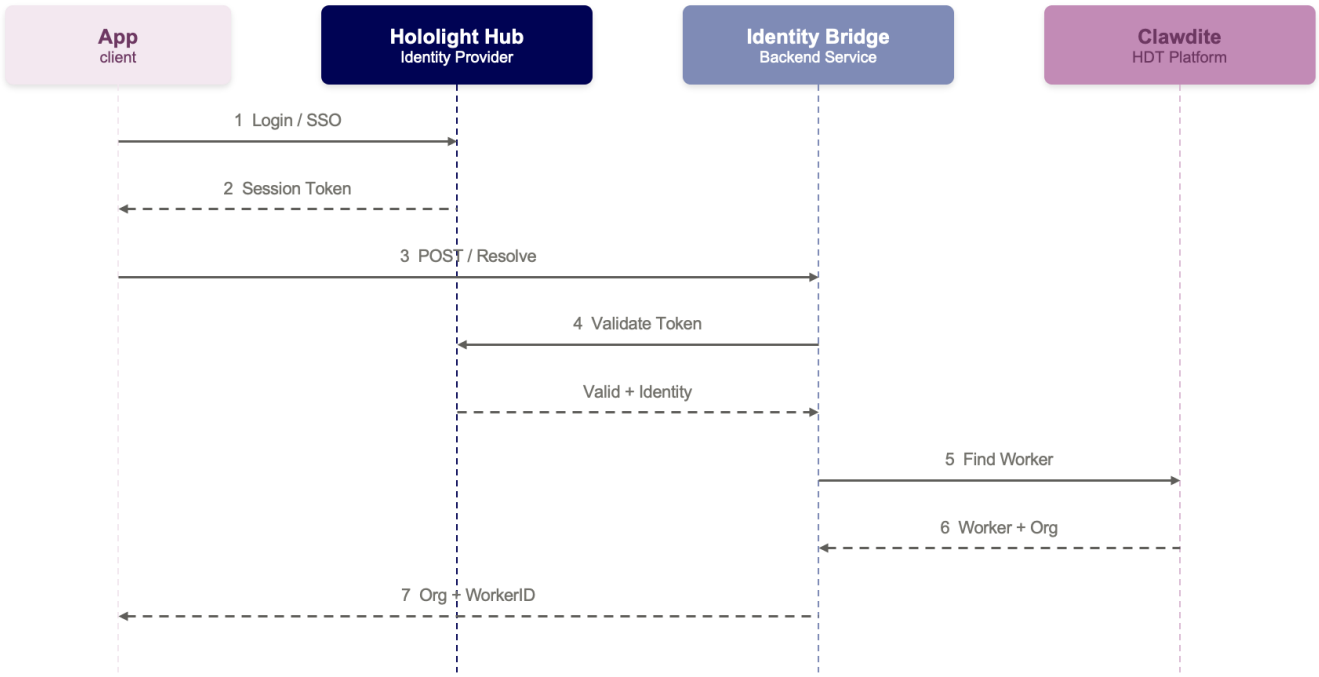


Figure 13 – Authentication and Identify-Resolution Flow linking XR Applications to the HDT

5.3.4 Identity-Mapping Mechanism

The identity-mapping mechanism ensures that the association between authenticated users and worker entities is maintained within the HDT, preserving it as the single source of truth.
This mechanism consists of two main phases:

- **Provisioning:** each worker entity in the Clawdite platform stores a reference to the corresponding identity used in Hololight Hub (e.g., email address or unique identifier) as a characteristic attribute.
- **Resolution:** during authentication, the Identity Bridge matches the identity claim extracted from the token with the stored attributes in the HDT, allowing retrieval of the corresponding worker entity.

This approach ensures traceability, consistency, and scalability, without requiring external identity mapping databases. It also supports full integration of authentication, user interaction, and adaptive XR personalisation within the XR5.0 ecosystem.

6 CONCLUSIONS

This deliverable has presented a comprehensive approach to personalised XR content development and adaptation within the XR5.0 project, addressing both methodological and technical challenges associated with human-centric adaptive systems.

One of the major outcomes of this work is the definition and validation of a structured framework for Adaptive User Interface design, which provides clear guidelines for developing XR applications capable of adapting to user characteristics and contextual factors. This framework establishes a strong foundation for future research and development in adaptive XR systems.

In addition, the deliverable introduces a novel approach to personalisation through the integration of Generative Artificial Intelligence. By combining implicit behavioural modelling with explicit user-driven inputs, the proposed system enables dynamic, real-time adaptation of training content. This represents a significant advancement over traditional rule-based approaches, offering greater flexibility, scalability, and user-centredness.

The development of the Adaptation Engine further contributes to the operationalisation of these concepts. As a modular and interoperable service, the engine enables seamless integration between XR applications, the training platform, and the human digital twin. Its design ensures that adaptive logic can be applied consistently across different environments while maintaining system scalability and robustness.

Another key achievement is the successful integration and validation of these components within XR5.0 pilot scenarios. The implementation of adaptive training in XR environments, together with stress monitoring and real-time interaction analysis, demonstrates the practical applicability of the proposed solutions in industrial contexts.

Overall, the work presented in this deliverable contributes directly to the XR5.0 vision of enabling human-centric, intelligent, and adaptive systems in Industry 5.0. The results provide a solid basis for further developments in subsequent work packages, particularly in terms of pilot validation, system optimisation, and large-scale deployment.

6.1 Lessons Learned

Several important lessons emerged from the work carried out in this deliverable.

First, the adoption of GenAI technologies significantly enhances the flexibility and effectiveness of adaptive systems. However, it also introduces challenges related to explainability, validation, and consistency, highlighting the importance of incorporating transparency mechanisms such as explainable AI interfaces.

Second, the integration of multiple components, including XR platforms, adaptation engines, and digital twin infrastructures, requires careful architectural design and clear definition of interfaces. Ensuring interoperability and data consistency is critical for achieving a cohesive and scalable system.

Third, human-centric design remains a fundamental requirement for adaptive XR systems. The combination of behavioural data and explicit user input proved essential for delivering meaningful and personalised experiences that accommodate diverse user profiles.

Finally, the validation of adaptive systems in realistic scenarios is crucial for assessing their effectiveness and usability. The pilot implementations provided valuable insights into user behaviour, system performance, and the practical challenges of deploying adaptive XR solutions in industrial environments.

These lessons will guide the future evolution of the XR5.0 platform and support the continued development of advanced, human-centric XR applications.

REFERENCES

[1] Matteri, D., Masiero, S., Dell'Oca, S., Montini, E., Cutrona, V., & Canetta, L., "A Protocol for Human-Centric Adaptive User Interfaces: From Static Interaction to Behaviour-Driven Adaptation," in *2025 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*, 2025.

[2] Canito, A., Mota, D., Marreiros, G., Corchado, J. M., & Martins, C., "Contextual adaptative interfaces for industry 4.0," in *In International Symposium on Distributed Computing and Artificial Intelligence*, 2020.

[3] Miraz, M. H., Ali, M., & Excell, P. S., "Adaptive user interfaces and universal usability through plasticity of user interface design," *Computer Science Review,* vol. 40, no. 100363, 2021.

[4] A. J. Chinchawade, "Beyond Static Interfaces: Real-Time Dynamic Ui Adaptation in Geosciences Using Large Language Models".

[5] Liu, Y.; Sra, M.; Xiao, C., "AlignUI: A Method for Designing LLM-Generated UIs Aligned with User Preferences," 2026.

[6] Colabianchi, S., Costantino, F., & Sabetta, N., "Assessment of a large language model based digital intelligent assistant in assembly manufacturing," *Computers in Industry,* vol. 162, no. 104129, 2024.