MOTIVATEXR

Maintenance, Support & Operation Training using Immersive Virtual and Augmented Technology for Efficiency with XR

D6.1 – CONTINUOUS INTEGRATION PLAN AND PLATFORM

10/12/2024

Funded by the European Union

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

Grant Agreement No.: 101135963
Call: HORIZON-CL4-2023-HUMAN-01-CNECT
Topic: HORIZON-CL4-2023-HUMAN-01-22
Type of action: HORIZON Innovation Actions

D6.1 - CONTINUOUS INTEGRATION PLAN AND PLATFORM

Work package	WP6
Task	T6.1
Due date	30/11/2024
Submission date	(10/12/2024) - Resubmission 30/01/2026
Deliverable lead	UPM
Version	V1.1
Authors	Francisco Moreno (UPM), Olga Chatzifoti (MAG)
Reviewers	Maritina Vlachaki (AAA), Gizem Senel (2F), Nikos Achilleopoulos (MAG)
Abstract	Continuous integration plan and platform. This deliverable presents the plans and tools deployed to ensure the continuous integration of the results of WP4 and WP5 to form the MOTIVATE XR releases.
Keywords	Integration, CI, Cybersecure Platform

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	12.11.2024	First TOC generated	Francisco Moreno (UPM)
V0.2	04.12.2024	First Complete Version	Olga Chatzifoti (MAG) Francisco Moreno (UPM)
V1.0	09.12.2024	Finalized after Review	Alexandra Malouta (AAA) Gizem Senel (2F) Olga Chatzifoti (MAG) Francisco Moreno (UPM) Nikos Achilleopoulos (MAG)
V1.1	06.03.2025	Extensions after V1.0	Olga Chatzifoti (MAG) Francisco Moreno (UPM) Nikos Achilleopoulos (MAG)

DISCLAIMER

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

COPYRIGHT NOTICE

PARTNERS

The Motivate XR Consortium is the following:

Participant number	Participant organisation name	Short name	Country
1	MAGGIOLI SPA	MAG	IT
2	CS GROUP-FRANCE	CS	FR
4	SOPRA STERIA GROUP	SOP	FR
5	F6S NETWORK LIMITED	F6S	IE
6	YOUBIQUO SRL	YBQ	IT
7	D-CUBE - NTI KIOUMP	D3	EL
8	2FREEDOM IMAGING SOFTWARE AND HARDWARE SL	2F	ES
9	CENTRO DI RICERCHE EUROPEO DI TECNOLOGIE DESIGN E MATERIALI	CETMA	IT
10	UNIVERSIDAD POLITECNICA DE MADRID	UPM	ES
11	TECHNISCHE UNIVERSITEIT DELFT	TUD	NL
12	FUNDACION TECNALIA RESEARCH & INNOVATION	TEC	ES
13	GORENJE GOSPODINJSKI APARATI, D.O.O.	HGE	SI
14	AEROSPACE VALLEY	AV	FR
15	BUILDING SYSTEMS INNOVATION CENTRE	AAA	EL
16	BI-REX- BIG DATA INNOVATION RESEARCH EXCELLENCE	BIR	IT
17	DIACHEIRISTIS ELLINIKOU DIKTYOU DIANOMIS ELEKTRIKIS ENERGEIAS AE	HEDNO	EL
18	AEROCAMPUS AQUITAINE	AC	FR

EXECUTIVE SUMMARY

This document outlines the Continuous Integration (CI) strategies implemented to ensure the integration of the platform and the integration of the outcomes from WP4 and WP5 of MOTIVATE XR project. By fostering collaboration and streamlining deployment processes, this plan ensures that MOTIVATE XR releases are robust, secure, and adaptable to the diverse needs of the ecosystem.

This deliverable consists of the following sections

- **Section 1 – Introduction:** Provides the purpose of the document, an overview of the MOTIVATE XR architecture, and highlights the benefits of continuous integration.
- **Section 2 – GitLab Repository Structure**: Describes a workshop conducted with the tool owners to understand their technical needs and their approach to integrating the tool within the platform
- **Section 3 – Environments**: Describes the key platform interactions and scenarios, including user access via the dashboard, third-party tool integration, API authentication, and internal service usage.
- **Section 4 - CI/CD Overview**: Outlines the stages of the CI/CD pipeline, including build, test, deploy, and publish, to ensure smooth integration and deployment.
- **Section 5 - GitLab CI Configuration:** Details the setup of GitLab CI, including runner configuration, gitlab-ci.yml, environment variables, and monitoring practices.
- **Section 6 - Automated Testing:** Emphasizes the importance of automated tests to maintain quality and reliability throughout the development process.
- **Section 7 - Tools Integration** Presents the methods and outcomes of integrating MOTIVATE XR tools, including API-based and dashboard-based options tailored to each tool's requirements.
- **Section 8 – Conclusions**: Provides an overview of the contents with projections about the next steps.

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Acronym	Title
API	Application Programming Interface
AR	Augmented Reality
BIM	Building Information Modelling
CAD	Computer-Aided Design
CD	Continuous Deployment
CI	Continuous Integration
CMS	Content Management System
CPU	Central Processing Unit
CRUD	Create Read Update Delete
CSRF	cross-site request forgery
GNSS	Global Navigation Satellite System
IdP	Identity Provider
MR	Mixed Reality
NAS	Network Attached Storage
NeRFs	Neural Radiance Fields
OAuth	Open Authentication
RTRX	Remote Training System with Augmented Reality
SDK	Software Development Kit
SEP	Semantic Processing Engine
UAT	User Acceptance Testing
VSLAM	Visual Simultaneous Localization and Mapping
VR	Virtual Reality
XR	Extended Reality
XSS	Cross-site Scripting

1 INTRODUCTION

Extended Reality (XR) is poised to revolutionize the way industries approach training, assistance, and operational workflows. By encompassing augmented reality (AR), virtual reality (VR), and mixed reality (MR), XR offers immersive experiences that can significantly enhance the effectiveness of industrial processes. This is particularly impactful in complex tasks such as assembly, manufacturing, maintenance, and dismantling of industrial goods, where traditional methods of training and assistance may fall short.

MOTIVATE XR aims to be a collaborative tool suited for authoring, publishing, and experiencing XR content. The platform is designed to facilitate collaboration among multiple stakeholders involved in industrial operations, providing tools for effective training and assistance. The ability to create, distribute, and manage XR content collaboratively ensures that industries can streamline their processes, improve safety, reduce errors, and enhance the overall efficiency of their operations.

To achieve this, a powerful Content Management System (CMS) is crucial for managing the XR content lifecycle. The CMS allows the platform to effectively organize, store, and distribute XR assets, ensuring that the right content is available at the right time for users and third-party tools. By leveraging a robust CMS, MOTIVATE XR provides a foundation that supports scalability, flexibility, and ease of use, enabling industries to harness the full potential of XR technologies in their workflows.

1.1 PURPOSE OF THE DOCUMENT

The purpose of this document is to outline a Continuous Integration (CI) and Continuous Deployment (CD) plan for the cybersecure MOTIVATE XR platform, that is used to manage and deploy applications. As this is the first version of the document, it will explain the workflow and of the CI/CD in a staging environment hosted by UPM and transitioning to a self-hosted Kubernetes environment. The final CI plan with the production environment will be provided on D6.2.

1.2 OVERVIEW OF THE MOTIVATE XR ARCHITECTURE

The MOTIVATE XR Cybersecure Architecture is a comprehensive platform designed to securely manage, store, and distribute XR files. It serves as a centralized hub that facilitates user interaction with XR content and provides access for third-party applications and tools.

The platform architecture is built around a containerized and customized open-source headless CMS. By using containerization, the platform ensures scalability and easy deployment across various environments. Integrated with a secure SQL database, a knowledge graph database and an Object storage, it efficiently stores and retrieves XR files and metadata. A public Application Programming Interface (API) with layered cybersecurity—including Open Authorization (OAuth) and API keys—will allow authorized external tools to interact safely with the CMS. For users, a dashboard

will provide a simple interface to manage XR content with real-time updates. Additionally, an Identity Provider (IdP) will handle authentication and authorization, ensuring only verified users and tools can access the platform. This setup aims to offer secure, scalable, and customizable content management that integrates smoothly with third-party tools. More detailed specifications of the Cybersecure Architecture are provided in the D3.5 "Functional Specifications & Cybersecure Architecture" document.

1.3 BENEFITS OF THE CI

Implementing a Continuous Integration and Continuous Deployment (CI/CD) pipeline provides significant advantages for MOTIVATEXR's development lifecycle, ensuring efficiency, quality, and collaboration. One of the primary benefits of CI/CD is automated testing, which guarantees that only code passing predefined quality checks is integrated into the main development branches, leading to potential reduction in the risk of introducing defects. This consistent validation process maintains the stability of the codebase and ensures that each update meets project standards.

Another key advantage of CI/CD is efficient Docker image management, where the process of building and pushing images is fully automated. This feature not only minimizes manual effort but also reduces the risk of errors, ensuring that every new version is reliably prepared for deployment. Additionally, the pipeline enables controlled releases, supporting structured release schedules and aligning deployments with project milestones and operational requirements.

The use of a CI/CD pipeline also enhances collaboration among developers by enforcing consistent branching and merging strategies. This structured approach fosters better coordination within the team, ensuring a smoother integration of contributions. Finally, the rigorous testing and deployment practices integral to CI/CD significantly improve code quality, minimizing the risk of bugs or regressions in production. These combined benefits make a CI/CD pipeline an essential tool for ensuring robust, efficient, and high-quality project delivery.

2 GITLAB REPOSITORY STRUCTURE

The UPM primarily utilizes GitLab as the platform for hosting and managing its code repositories. In line with this practice, the MOTIVATEX R's Cybersecure Development project will also leverage GitLab's robust infrastructure, allowing all related repositories to be securely hosted in private repositories.

2.1 BRANCHING STRATEGY

MOTIVATE XR employs a structured branching strategy within GitLab to streamline collaboration and uphold high standards of code quality. This strategy revolves around two main branches, main and dev, along with feature-specific and bug-fix branches that promote a clean and organized workflow.

2.1.1 MAIN BRANCH

The main branch serves as the production-ready codebase and represents the stable version of the application currently deployed. This branch is protected to ensure it remains free from untested or unstable changes. Updates to the main branch occur only when a new release has been thoroughly validated and approved, ensuring that production always reflects reliable and high-quality code.

2.1.2 DEV BRANCH

The dev branch acts as the primary integration branch where active development takes place. Developers create individual branches for new features or bug fixes, branching off from the dev branch. The dev branch is periodically prepared for a new release. At this stage, the dev branch undergoes comprehensive testing, which includes automated tests, peer code reviews, and deployment to a staging environment, if necessary. These tests simulate production-like conditions to identify and resolve potential issues before release. Once the dev branch satisfies all quality standards, it is merged into the main branch, marking the code as production-ready and eligible for deployment.

This structured approach ensures a clear separation between the development and production environments, providing multiple benefits that enhance both efficiency and reliability. By fully validating changes before deployment, this approach significantly reduces the risk of introducing instability into the production environment. Additionally, it simplifies the tracking and management of changes, making it easier to implement rollbacks when necessary. Furthermore, isolating individual contributions until they are ready for integration fosters a more collaborative development process, enabling developers to work independently while maintaining the integrity of the shared codebase.

2.1.3 FEATURE/HOTFIX BRANCHES

For every new feature or bug fix, a dedicated branch is created from the dev branch to isolate new features or bug fixes. These branches are named descriptively to indicate their purpose, such as *feature/user-authentication* or *bugfix/login-error*. By isolating work into dedicated branches, developers can focus on their tasks without risking interference with the main development workflow. Once a feature or fix is completed, tested locally, and reviewed, it is merged back into the dev branch. This process ensures that incomplete or unstable changes do not impact the overall progress of development.

2.2 MERGE POLICIES AND RULES

To maintain code quality and ensure stability, the project enforces strict merge policies and rules.

2.2.1 CODE REVIEW REQUIREMENTS

Before any branch is merged into the dev or main branches, the code must undergo a thorough peer review. This process requires at least one other developer to review the changes, to ensure they comply with the project's coding standards, are well-documented, and do not introduce any new issues. Code reviews promote knowledge sharing, improve code quality, and help identify potential problems early in the development process.

2.2.2 AUTOMATES TESTING PREREQUISITES

Automated testing is a critical component of the merge process. Before merging, all changes must pass a suite of automated tests, which may include unit tests, integration tests, and other quality checks. If any test fails, the merge is blocked until the issues are resolved. This process ensures that only code that meets predefined quality standards is integrated into the dev or main branches.

2.2.3 MERGING RULES

	Merging into dev	Merging into main
Prerequisites	Code review approval from at least one other developer.	Comprehensive testing on the dev branch, including automated tests and possibly staging deployments to validate the new code under conditions that mimic production.
	All automated tests pass successfully.	All code review comments are addressed.

Process	Developer submits a merge request from the feature/hotfix branch to the dev branch.	A merge request is created from the dev branch to the main branch.
	The CI/CD pipeline runs automated tests.	Final verification is performed, ensuring all quality standards are met.
	After passing tests and receiving code review approval, the merge is completed.	Upon approval, the merge is completed.
	The version number is incremented appropriately (minor or patch).	The major version number is incremented.

TABLE 1 MERGING RULES

2.3 VERSIONING

MOTIVATE XR platform adopts a standardized versioning strategy based on Semantic Versioning (SemVer) to maintain clarity, consistency, and predictability throughout the development lifecycle. This approach ensures that all stakeholders have a clear understanding of the scope and impact of changes made to the software.

2.3.1 SEMANTIC VERSIONING (SEMVER)

Semantic Versioning follows the format **MAJOR.MINOR.PATCH**, where each segment of the version number signifies the nature of the changes:

- **MAJOR version (X.0.0):** Incremented for incompatible API changes or significant modifications that may not be backward-compatible.
- **MINOR version (0.Y.0):** Incremented when new features or enhancements are added in a backward-compatible manner.
- **PATCH version (0.0.Z):** Incremented for backward-compatible bug fixes or **Version Increment Rules.**

Patch Version Increment (X.Y.Z → X.Y.(Z+1))

- **When to Increment:**
  - Fixing bugs or defects that do not affect the API or functionalities.
  - Making minor improvements or optimizations.

- **Examples:**
  - Correcting a minor bug that doesn't alter application behaviour significantly.
  - Making small performance enhancements.

Minor Version Increment (X.Y.Z → X.(Y+1).0)

- **When to Increment:**
  - Adding new features or enhancements that are backward-compatible.
  - Introducing new functionalities without breaking existing APIs.
- **Examples:**
  - Implementing a new feature module.
  - Adding new endpoints to an existing API.

Major Version Increment (X.Y.Z → (X+1).0.0)

- **When to Increment:**
  - Making changes that are not backward compatible.
  - Removing or significantly altering existing functionalities.
  - Overhauling system architecture or core components.
- **Examples:**
  - Refactoring the application in a way that changes existing API behaviours.
  - Eliminating deprecated features that clients might still use.

2.3.2 VERSIONING ACROSS BRANCHES

To maintain consistency and clarity throughout the development process, a versioning strategy is applied across different branches in a structured manner. This makes sure that each branch accurately reflects the state of the code and its readiness for deployment.

- **Feature/Hotfix Branches:**
  - Utilize pre-release identifiers (e.g., **1.3.0-beta.1**) during development.
  - Helps in tracking progress and testing pre-release versions.
- **Dev Branch:**
  - **MINOR** and **PATCH** versions are incremented upon successful merges of features or bug fixes.
  - Reflects the cumulative changes ready for staging and further testing.
- **Main Branch:**
  - **MAJOR** version is incremented when merging into the main branch for a production release.
  - Signifies a release with significant updates or potential breaking changes.

2.3.3 TAGGING AND RELEASE PROCESS

An organized tagging and release process is essential for tracking changes, facilitating rollbacks, and communicating updates to stakeholders. This strategy incorporates systematic tagging and detailed release notes to effectively document each version.

- **Docker Tagging:**
  - Each release is tagged in the Docker repository using the version number. **Git Tagging:**
  - Each release is tagged in the Git repository using the version number.
- **Release Notes:**
  - Document the details of each release, categorized by:
    - New Features
    - Bug Fixes
    - Breaking Changes
    - Deprecations
  - Provide migration guides, if necessary, especially for major releases.

2.3.4 INTEGRATION WITH CI/CD PIPELINE

Integrating versioning strategy with the CI/CD pipeline automates version management and enforces consistency across the development process. This integration enables version increments and changelogs to be handled systematically, reducing manual effort and potential errors.

- **Automated Version Bumping:**
  - The CI/CD pipeline automates version number increments based on commit messages or merge actions.
  - Uses tools or scripts to update version numbers consistently.
- **Changelog Generation:**
  - Automatically generates or updates the changelog with details from commit messages or pull requests.
  - Ensures transparency and keeps all stakeholders informed.
- **Compliance Checks:**
  - The pipeline includes checks to ensure version numbers follow SemVer rules.
  - Prevents accidental releases without proper version increments.

3 ENVIRONMENTS

MOTIVATE XR project employs a multi-tiered development environment strategy to maintain code quality, facilitate thorough testing, and streamline the deployment process. This approach involves three primary environments—Local, Staging, and Production—each serving a specific purpose in the development lifecycle and allowing for a controlled and efficient progression of code from inception to deployment.

3.1 LOCAL ENVIRONMENT

The Local Environment is where individual developers write, compile, and test their code on their personal machines. In this setting, developers have the freedom to experiment with new features, fix bugs, and perform initial testing without impacting the shared codebase. They can customize their development setup with tools and configurations that enhance productivity and run unit tests to validate the functionality of individual components before integration. Working in isolated local environments enables developers to iterate rapidly, catch errors early, and verify their code meets initial quality standards before sharing it with the team.

3.2 STAGING ENVIRONMENT

Serving as a critical intermediary between development and production, the Staging Environment closely replicates the production environment. It provides a platform to simulate production conditions by mirroring hardware, software, and network configurations, which helps in identifying environment-specific issues. In the staging environment, code from the dev branch, including the latest merged features and fixes, is deployed. This environment allows for comprehensive testing, including integration testing to make sure different modules and services work together, system testing to validate the complete and integrated software product, and User Acceptance Testing (UAT), where stakeholders verify that the system meets business requirements. The staging environment is essential for identifying and resolving issues such as bugs, performance bottlenecks, or security vulnerabilities that may not be apparent in the local environment. By serving as the final testing ground, it ensures that code is production-ready and reduces the risk of issues upon deployment.

UPM will provide this environment, a Kubernetes cluster with three nodes, which will try to closely mirror the production setup to provide an accurate testing ground for the platform. Each node is provisioned with ample Central Processing Unit (CPU) and memory resources to effectively handle the anticipated load during testing phases. The Kubernetes cluster includes essential components such as an Ingress controller to manage external access. It uses ConfigMaps and Secrets for dynamic configuration and sensitive data management. A Network Attached Storage (NAS) has been integrated for persistent storage solutions using Persistent Volume Claims (PVCs) to support stateful applications and data persistence across deployments. Monitoring and logging tools like Prometheus and Grafana could be installed, enabling real-time performance tracking and

facilitating prompt identification and resolution of issues. HTTPS with SSL/TLS protocols will be implemented throughout the environment. This setup enables encrypted connections between clients and services, safeguarding data in transit. SSL/TLS certificates are securely managed and integrated with the Ingress controller to terminate HTTPS connections, providing secure external access to the services running within the cluster.

3.3 PRODUCTION ENVIRONMENT

The Production Environment is the live system where the application is available to end-users. Hosting the code from the main branch, this environment is characterized by stability and reliability, as only thoroughly tested and approved code is deployed to guarantee a dependable user experience. Strict access controls are in place to prevent unauthorized changes, maintaining the integrity of the live application. Continuous monitoring and maintenance are critical in this environment, with performance monitoring to detect and address issues proactively and robust security measures to protect user data and system integrity. The deployment process is carefully controlled, with manual release triggers allowing for oversight and coordination, and rollback capabilities available in case critical issues are discovered. The production environment represents the culmination of the development process, delivering value to users and reflecting the project's success. Details of the production environment will be included in the Deliverable D6.2.

4 CI/CD OVERVIEW

After code changes are merged into the dev or main branch, the Continuous Integration and Continuous Deployment (CI/CD) pipeline is automatically triggered to streamline the integration of new features and maintain high code quality. This workflow is composed of four critical stages: Build, Test, Deploy, and Publish. The first three stages are triggered by merging into the Dev branch, while the publish stage is only triggered when merging into the main branch. Each stage plays a vital role in ensuring that the application is thoroughly tested, securely packaged, and efficiently deployed to both the staging and production environments.

4.1 BUILD STAGE

The Build stage initiates the CI/CD pipeline by constructing Docker images based on the latest code from the dev branch. This process involves compiling the application code and encapsulating it within Docker containers, along with all necessary dependencies and environment configurations. By containerizing the application, the consistency is achieved across different deployment environments, reducing environment problems. The Docker images serve as the standardized units for subsequent testing and deployment phases, ensuring that the same artifact is tested and eventually deployed to production.

4.2 TEST STAGE

Following the successful build of Docker images, the pipeline progresses to the **Test** stage. Here, a comprehensive suite of automated tests is executed to validate the integrity and functionality of the application components.

The Test stage is crucial for detecting bugs, regressions, or any unintended side effects introduced by recent code changes. If any tests fail, the pipeline halts, and developers are notified to address the issues before proceeding. This gatekeeping mechanism maintains the overall quality and stability of the codebase.

4.3 DEPLOY STAGE

After successfully passing all tests, the pipeline proceeds to the Deploy stage. In this stage, the newly built and tested Docker images are retrieved from a registry, such as Docker Hub or GitLab's Container Registry, and deployed to the Kubernetes Staging Environment by pulling the latest Docker images.
Deploying to staging enables the team to validate deployment scripts and configurations, making sure that Kubernetes functions correctly. It also provides an opportunity to perform manual and exploratory testing, identifying any issues that automated tests might have missed. Additionally, the team can conduct performance and load testing to allow external tools to verify their integration with the platform. This stage is essential for catching any environment-specific issues and providing a final verification before the application is promoted to production.

4.4 PUBLISH STAGE

The final stage involves updating the Production Environment with the latest application version. The CI/CD pipeline triggers a rolling update on the production Kubernetes cluster, deploying the newly built Docker images. Kubernetes orchestrates the update by replacing the old pods with new ones running the updated application, ensuring minimal downtime transition.

5 GITLAB CONFIGURATION

The Continuous Integration and Continuous Deployment (CI/CD) pipeline is orchestrated using GitLab CI/CD, configured through the .gitlab-ci.yml file located at the root of the repository. This configuration defines the stages, jobs, and scripts that automate the building, testing, deployment, and publishing of the application. The pipeline ensures that code changes are systematically processed and deployed, maintaining consistency and reliability across environments.

5.1 GITLAB RUNNER CONFIGURATION

To execute the CI/CD pipeline, a self-hosted GitLab Runner is utilized within the environment. This runner is installed on dedicated servers and is responsible for running the pipeline jobs defined in the .gitlab-ci.yml file. By hosting the runner internally, greater control over the execution environment is gained, including:

- **Security**: All code and artifacts remain within the infrastructure, reducing exposure to external threats.
- **Customization**: The runner environment can be tailored with specific tools, dependencies, and configurations required for the projects.
- **Resource Management**: Resources such as CPU, memory, and storage can be allocated according to needs, ensuring optimal performance.

The runner is registered with the internal GitLab instance using a unique token and is tagged appropriately (e.g., motivatexr-runner). This tag is used in the CI configuration to direct jobs to the self-hosted runner. The runner is configured to run with the Docker executor, allowing each job to run in an isolated Docker container, which enhances security and consistency.

5.2 GITLAB-CI.YML

The gitlab-ci.yml file is structured into the four primary stages defined on section 4 CI/CD Overview. Each stage contains specific jobs that execute sequentially, ensuring that the application is thoroughly built, tested, and deployed.

5.2.1 BUILD STAGE

In the Build stage, the job build compiles the application and builds Docker images. The script executes commands to build the application and package it within Docker containers. After building, any running Docker Compose services are shut down to clean up the environment.

5.2.2 TEST STAGE

The Test stage contains the job test, which runs automated tests to validate the application's functionality. The application is started in a development environment using Docker Compose, and tests are executed inside the Docker container. Upon completion, Docker Compose services are shut down.

5.2.3 DEPLOY STAGE

In the Deploy stage, the job deploys handles the deployment to the Kubernetes Staging Environment. The script starts the application, logs into the Docker registry securely using

environment variables (e.g., DOCKER_PASSWORD), and pushes the Docker images to the registry. After pushing the images, Docker Compose services are shut down on the runner, and finally the Kubernetes staging cluster is triggered to update its pods with the latest images.

5.2.4 PUBLISH STAGE

The Publish stage includes the job publish, responsible for deploying the application to the Production Environment. This stage triggers the Kubernetes production cluster to update its pods, ensuring that the latest version of the application is running in production.

5.3 EXAMPLE GITLAB-CI.YAML FILE

Below is an example of the .gitlab-ci.yml file, with scripts represented in pseudocode to focus on the structure and flow of the pipeline:

```
stages:
 - build
 - test
 - deploy
 - publish
build:
 stage: build
 script:
  - # Build the application and create Docker images
  - # Shut down any running Docker Compose services
 only:
  - dev
 tags:
  - motivatexr -runner

test:
 stage: test
 script:
  - # Start the application in a development environment using Docker Compose
  - # Execute automated tests inside the Docker container
  - # Shut down Docker Compose services after testing
 only:
  - dev
```

```
 tags:
  - motivatexr -runner

deploys:
 stage: deploy
 script:
  - # Start the application
  - # Log in to the Docker registry securely
  - # Push the Docker images to the registry
  - # Shut down Docker Compose services
  - # Trigger the Kubernetes Staging Environment to update pods
 only:
  - dev
 tags:
  - motivatexr -runner

publish:
 stage: publish
 script:
  - # Trigger the Kubernetes Production Environment to update pods
 only:
  - main
 tags:
  - motivatexr -runner
```

The pipeline is designed to automate the workflow after merging changes into the dev branch:

- **Branches**: The only: - dev directive ensures that the pipeline runs only when changes are pushed to the dev branch, aligning with our branching strategy. Meanwhile the – main directive ensures that the pipeline runs only when changes are pushed to the main branch
- **Tags**: Each job specifies tags: - motivatexr-runner, directing the jobs to run on the self-hosted GitLab Runner.
- **Scripts**: The script section in each job contains the commands executed during that stage. These scripts are written in pseudocode for clarity but include actions such as building Docker images, running tests, logging into the Docker registry, pushing images, and triggering Kubernetes updates.

5.4 ENVIRONMENT VARIABLES AND SECURE CREDENTIALS

Sensitive data such as passwords and tokens are managed securely using GitLab's CI/CD variables. These variables are stored in the project's settings and are injected into the runner's environment at runtime. This approach confirms that sensitive information is not hard coded into the scripts or exposed in the repository.

5.5 NOTIFICATIONS AND MONITORING

Notifications can be set up to be sent to the development team to keep everyone informed about the pipeline status. These notifications, delivered via email or messaging platforms like Slack or Discord, provide real-time updates on the progress and outcomes of each stage. These notifications can be triggered after the successful completion of stages such as Deploy or Publish or by not passed test on Test stage or errors on the Deploy stage. Additionally, Gitlab provides tools to monitor the health and performance of the applications in both the staging and production environments.

6 AUTOMATED TEST

To check the reliability and robustness of our application architecture, which includes a headless CMS, an Identity Provider (IdP) for identity and access management, a database for data persistence and a Dashboard for end-users, a comprehensive suite of automated tests will be implemented. These tests will cover various aspects of the system to detect issues during the development cycle and maintain high-quality code throughout the project's lifecycle.

6.1 UNIT TESTS

Unit Tests will be written for individual components within the headless CMS, such as controllers, services, and models. These tests will verify that each unit of code functions correctly in isolation. For example, automated tests will verify that the CMS correctly handles all CRUD (Create, Read, Update, Delete) operations for XR content, ensuring data integrity and consistency over individual API endpoints. In addition, unit tests will be created for any custom integrations with the IdP, verifying that authentication tokens are correctly processed, and that user roles and permissions are accurately interpreted within the CMS. For the dashboard, unit test for components and methods, must be created as well.

6.2 INTEGRATION TESTS

Integration Tests will focus on the interaction between the IdP, the CMS and the database. These tests will simulate real-world scenarios where multiple components work together, such as user

authentication flows, authorization checks, and data retrieval operations. The end-to-end authentication process should be tested, to ensure that users can log in via IdP and access protected resources in the CMS based on their assigned roles. Integration tests will also verify that data is correctly stored in and retrieved from the database when performing CRUD operations through the CMS API. The interaction between the frontend and CMS should be tested, verifying that API calls return the expected data, and that the dashboard handles responses correctly.

6.3 DASHBOARD SPECIFIC TESTS

UI tests will verify that the visual elements of the dashboard render correctly across different devices and screen sizes. Automated security testing will detect vulnerabilities in the dashboard, test like cross-site scripting (XSS) or cross-site request forgery (CSRF).

7 TOOLS INTEGRATION WORKSHOP #1

To gather the functional requirements for the CMS platform, a dedicated workshop was conducted during a technical meeting. The workshop was attended by representatives from seven MOTIVATE XR tools that plan to utilize the CMS for storing or retrieving XR content and for authentication purposes. The primary goal was to establish how the tools were going to integrate or interact with the platform. The workshop lasted one hour and was structured into the following key segments.

7.1 PRESENTATION OF PLATFORM CAPABILITIES

The session began with a comprehensive presentation highlighting the various ways the CMS platform could support the tools. This included an overview of the platform's functionalities such as content management, authentication services, data processing capabilities, and integration options. The presentation aimed to provide participants with a clear understanding of how the platform could meet their specific needs.

7.2 INTERACTIVE MIRO BOARD SESSION

Following the presentation, an interactive Miro board was introduced to facilitate collaborative input. Each tool developer was provided with a template on the Miro board to specify their needs and requirements. The template included the following sections:

- **Tool Name and Description:** *Provide the name of your tool and a brief description of its functionality.*
- **Integration Method:** *Do you prefer integration via API, software development kit (SDK), webhooks, or other methods?*

- **Output/Input Expected:** *Is the data flow one-way or two-way between your tool and the CMS?*
- **Supported Data Formats:** *Which data formats does your tool support (e.g., JSON, XML, CSV, XR-specific formats)?*
- **Authentication and Security:** *Do you need to secure your API or authorization for your tool?*
- **Features Needed from CMS:** *List specific CMS functionalities you need access to (e.g., data cleaning, data augmentation).*
- **User Roles and Permissions:** *What levels of access will your tool require within the CMS?*
- **Monitoring and Analytics:** *Do you need access to analytics or logs related to the integration?*
- **Notification Preferences:** *Would you like to receive notifications for certain events (e.g., data updates, errors)?*
- **Other:** *Any other need or requirement your tool might have.*

This interactive session allowed each participant to articulate their specific requirements and preferences in a structured manner.

7.3 DISCUSSION AND FEEDBACK

After filling out their respective sections on the Miro board, the participants engaged in an open group discussion. This segment provided an opportunity to:

- **Discuss Integration Strategies:** The discussion facilitated an exchange of ideas on how best to integrate the tools with the CMS, considering factors like preferred data formats and authentication methods.
- **Clarify Requirements:** Participants elaborated on their inputs, providing additional context where necessary.
- **Identify Common Needs:** The group identified overlapping requirements and potential areas for shared solutions.
- **Address Potential Challenges:** Participants brought up any concerns or potential obstacles they foresaw in the integration process, allowing for proactive problem-solving.

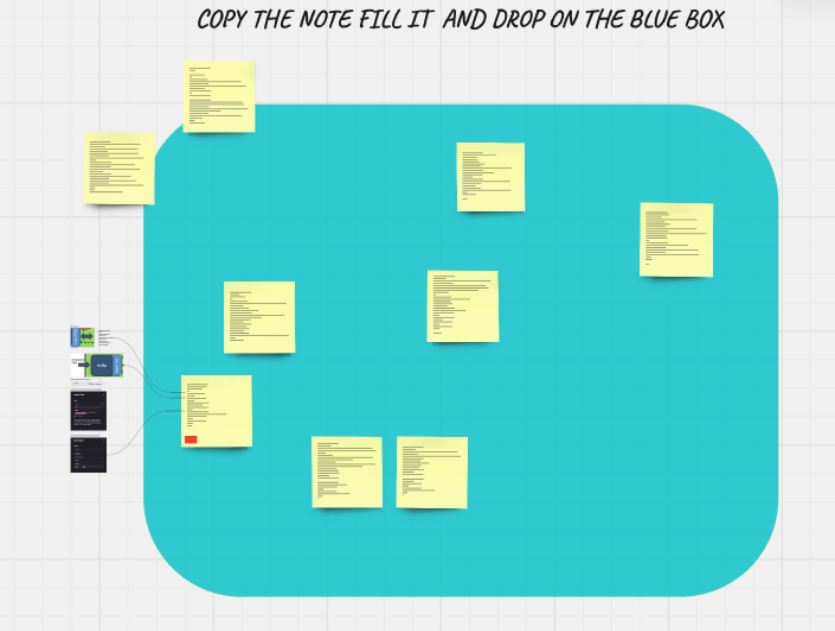

FIGURE 1 MIRO BOARD FROM WORKSHOP

7.4 GENERAL OUTCOMES

The workshop successfully achieved its objectives:

- **Use of the platform:** Identified four different ways to interactor integrate the CMS on the different tools.
- **Integration approach:** Retrieved the first approach on how each tool will integrate the CMS.
- **Requirements Gathering:** Collected detailed functional requirements from each tool, providing valuable insights into the specific needs of the users.
- **Enhanced Collaboration:** Fostered a collaborative environment where participants could share ideas and build upon each other's inputs, strengthening the community around the platform.
- **Actionable Insights:** The feedback and discussions generated actionable insights that will inform the next steps in the platform's development, ensuring it aligns closely with user needs.

7.5 TOOLS INPUTS

The following outlines the input provided by each tool, from the filled templates.

7.5.1 KAYROX

Tool Name and Description	KAYROX
Integration Method:	API
Output/Input expected:	two-way, read and write content: media assets, experiences (json+assets)
Supported Data Formats	JSON, XML, CSV- 3d models: GLB, FBX - Images: jpg png svg- Video: mp4 webm - PDFs, Audio: mp3 ogg
Authentication and Security	Yes (registered user with login&password; https protocols...)
Features Needed from CMS	Single sign-on with identity management Asset Storage
User Roles and Permissions:	A user always belongs to an organization (company). A user can have one of 3 roles: -Consumer: Can only consume experiences -Creator: Can create and consume experiences -Administrator: Can do the same as a Creator but also manages aspects of the Organization such as creating new Users. In addition, it is possible to publish experiences at a public level. These can be consumed without the need for a username and password.
Monitoring and Analytics:	Yes, especially if the plan of exploitation is to be pay-per-use.

Notification Preferences:	Yes, i.e. when a video to 3D process has finished, or a document has been processed to a KG
Other	AI services, Video to 3D

TABLE 2 KAYROX WORKSHOP OUTCOME

7.5.2 RTXR

Tool Name and Description	Remote Training System with Augmented Reality (RTXR)
Integration Method:	API or Via Dashboard
Output/Input expected:	Two-Way, read and write content
Supported Data Formats	JSON, 3D Models: FBX, Images: jpg, png, Videos: mp4 Audio: mp3
Authentication and Security	Server-side authorization ideally would be nice
Features Needed from CMS	Discuss further
User Roles and Permissions:	Need to be defined, Content creator, trainer-trainee, remote assistant.
Monitoring and Analytics:	Yes, would be nice.

Notification Preferences:	Yes, for data updates, (NE live function), training material uploaded, etc.
Other	Not at the moment

TABLE 3 RTXR WORKSHOP OUTCOME

7.5.3 MIRA

Tool Name and Description	MIRA - Digital twin platform
Integration Method:	API
Output/Input expected:	One way
Supported Data Formats	Json, csv
Authentication and Security	Already implemented.
Features Needed from CMS	Discuss
User Roles and Permissions:	Existing roles in Mira relate to 1) owner 2) admin 3) user
Monitoring and Analytics:	Likely no
Notification Preferences:	Likely no
Other	-

TABLE 4 MIRA WORKSHOP OUTCOME

7.5.4 INSCAPE VTS EDITOR

Tool Name and Description	Inscape VTS Editor Authoring tool used to create design and publish an XR experience in internal format usable by Inscape VTS Player software. we have internal collaborative work capabilities.
Integration Method:	it's a software to be run on a station and interactively used by the user (an editor), how do you integrate this ? via Dashboard
Output/Input expected:	Input : UE data from 3D models, IA extracted data / output : export XR experience (Inscape format), partial XR data export (generated for KAYROX)
Supported Data Formats	JSON, internal Inscape format
Authentication and Security	not sure, maybe
Features Needed from CMS	File storage and exchange, file versioning
User Roles and Permissions:	Author (creation, edition), trainer (edition)
Monitoring and Analytics:	not sure
Notification Preferences:	I don't see how we could manage them, so far
Other	...

TABLE 5 INSCAPE VTS EDITOR WORKSHOP OUTCOME

7.5.5 INSCAPE VTS PLAYER

Tool Name and Description	Inscape VTS Player: Application used to play an XR experience in internal format on display devices
Integration Method:	t's an application to be run on a station with a connected XR display, or run directly on the XR display, it needs to retrieve the XR experience files from storage.
Output/Input expected:	input : XR experience (Inscape format)
Supported Data Formats	internal Inscape format
Authentication and Security	not sure, maybe
Features Needed from CMS	Retrieval of XR experience files from data storage.
User Roles and Permissions:	trainee (playing)
Monitoring and Analytics:	not sure
Notification Preferences:	I don't see how we could manage them, so far
Other	...

TABLE 6 INSCAPE VTS PLAYER

7.5.6 3D VIDEOGRAMMETRY SYSTEM

Tool Name and Description	-"**XVS Scanner"** Hardware is a 3D Scanning tool that can be used to scan areas by utilising the XVS Scanner. Other Hardware can be used like glasses or manual RGBD sensors, LiDAR.. **- Videogrammetry Software** allows to enter videos and obtain 3D models. AI-based Segmentation 3D available. 3D processing in our Servers.
Integration Method:	API
Output/Input expected:	XVS App today: Input: mp4,2fc // Output:las, ply, obj, Others also are possible.
Supported Data Formats	mp4, las, ply, obj, stl
Authentication and Security	Actually, it has a license code. Can be adapted.
Features Needed from CMS	Not sure
User Roles and Permissions:	Not Needed
Monitoring and Analytics:	Not Sure
Notification Preferences:	Not Sure

Other	Not at the moment

TABLE 7 3D VIDEOGRAMMETRY SYSTEM WORKSHOP OUTCOME

7.6 INTEGRATION OPTIONS

After analyzing the discussions and inputs gathered during the workshop, multiple methods of integration with the platform have been identified. By defining these use cases, the goal is to illustrate the platform's versatility and demonstrate how it effectively covers all the specific needs faced by MOTIVATE XR tools.

7.6.1 ACCESS VIA DASHBOARD

This integration method is designed for tools or applications that do not wish to integrate directly with the platform using the API. The dashboard offers an alternative method of interaction. Users can export their XR files from their tools and manually import them into the dashboard, and vice versa. This process provides flexibility, allowing users to continue utilizing their existing tools while still benefiting from the platform's centralized management and collaboration features.

The platform will offer a dashboard designed for managing assets, XR files and experiences. Accessible directly through web browsers, this dashboard provides a unified interface for all XR content management needs.

Aimed at individual users or teams who design and develop XR content—such as 3D artists, developers, and designers—the dashboard also serves team members, clients, or partners who require access to XR files for review, feedback, or contribution. This inclusive approach makes it easy to collaborate among all stakeholders involved in the XR creation process.

Users can log in to the dashboard via their web browser on any device using secure credentials, ensuring that their work remains protected. Once logged in, they can manage their XR files directly from the intuitive dashboard interface. This includes uploading new files, organizing assets efficiently or modify and updating metadata associated with their XR files easily.

By securing their work within the platform and making it accessible everywhere and anytime, users benefit from increased productivity and peace of mind. The dashboard's user-friendly design and powerful features streamline the XR content management process, empowering creators and collaborators to focus on bringing immersive experiences to life.

7.6.2 ACCESS VIA API

The platform will offer integration capabilities for third-party tools and applications, enabling MOTIVATE XR stakeholders to incorporate XR content management functionalities directly into

their own solutions. By utilizing the platform's APIs, these external tools will be able to perform many of the same actions available on the platform's dashboard, such as uploading XR files, editing metadata, and more, all within their native environments.

This integration is designed for tool creators and developers who wish to integrate their applications with MOTIVATE XR. By connecting to the platform, they can focus on their tools while providing users with access to XR content management features.

Developers will have two methods to integrate their tools with the platform, via Identity Provider (IdP) or via API Key Authentication.

7.6.2.1 IDENTITY PROVIDER

Developers can integrate their tools using OAuth through the IdP. This method involves redirecting users to a secure login and authorization process, where they grant the tool permission to access their data on the platform.

- **Use Cases:** Best suited for tools that need to perform actions on behalf of a user, such as personal content management, editing, or when accessing user-specific data.
- **Access Control:** OAuth provides user-authenticated endpoints, allowing the tool to perform any actions that the authenticated user is permitted to do on the platform.
- **Security Considerations:** OAuth ensures that users' credentials are not shared with the tool. Instead, the tool receives a token with specific permissions granted by the user, enhancing security and compliance with privacy standards.

7.6.2.2 API KEY AUTHENTICATION

Developers can authenticate their tools using API Key Authentication. In this method, the platform issues a unique API key to the developer, which the tool uses to authenticate its requests to the platform's API. This key acts as a credential that identifies the tool and grants it access to specific API endpoints appropriate for its designated role.

- **Use Cases:** Ideal for server-to-server interactions, automated processes, or when the tool operates independently of a user's direct input.
- **Access Control:** The API key provides access to endpoints suitable for the tool's functions, ensuring that it can perform necessary actions without overstepping permissions.

- **Security Considerations:** Developers must securely store and manage the API key to prevent unauthorized access. The key should be kept confidential and rotated periodically according to best practices.

7.7 INTEGRATION METHOD BY TOOL

In the initial version of this document, the most suitable integration method for each tool in the MOTIVATE XR ecosystem was presented. Some tools integrate with the platform via API Key Authentication, enabling secure server-to-server communication and automated processes with controlled permissions. Others utilize OAuth Authentication through an Identity Provider, allowing users to securely grant specific permissions for tools that perform actions on their behalf. Certain tools may prefer not to integrate directly with the platform's API and instead use the dashboard method—manually exporting content from their tool, importing it into the dashboard, and vice versa. In the previous version of the document, each tool's integration was detailed in this section; however, as both the integrations and the tools themselves have evolved, the various integration methods are now presented in Section 8.1.

8 TOOLS INTEGRATION WORKSHOP #2

This second workshop was designed as a follow-up to the initial session, focusing on reviewing and aligning the integration flows of each user with the platform, following recent architectural changes. Since all participants were already familiar with the platform, the goal was to share and understand the updated integration processes and gather feedback on potential improvements.

Following the discussions and feedback gathered during the second workshop, we decided to reorganize and redefine the set of tools that make up the platform. As a result, we introduced a new categorization that clearly separates the tools into four main groups: Asset Creation Tools, XR Authoring Tools, AI Services, and Experience Tools. This new disposition aims to simplify the overall architecture, clarify the role of each component, and improve the integration process for users. Additionally, some tools have been renamed, merged, or significantly reworked to better align with this updated structure and to address overlapping functionalities or gaps identified during the workshop.

Prior to the workshop, all participants were asked to prepare and bring a diagram representing the flow of their integration with the platform. These diagrams were to be drawn directly on a shared Miro board, allowing for real-time discussion and adjustments. Each participant presented their flow diagram to the group. Together, we reviewed each integration step by step, discussing potential issues, clarifying uncertainties, and identifying opportunities for better alignment with the updated architecture and we came up with the following integration flows.

8.1 ASSETS CREATION TOOLS

These tools are a set of applications for the creation of assets such as digital twins and 3D models, which can be used across different authoring and experience tools within the MotivateXR environment.

8.1.2 3D VIDEOGRAMMETRY SYSTEM

Tool	3D Videogrammetry System
Description	3D Scanning and Modelling Tool for XR Devices
Integration Method	API Key and Dashboard
Responsible	2Freedom

TABLE 8 3D VIDEOGRAMMETRY SYSTEM INTEGRATION

The 3D Videogrammetry System allows users to generate 3D models from video footage captured either with a standard camera or with 2Freedom XVS Scanner devices.

When using a standard camera, the user first accesses the MotivateXR web application, logs in through the platform's Identity Provider (IdP), and selects an existing or creates a new project. The user then uploads the video footage to the MotivateXR Content Management System (CMS). When upload the user triggers the 2Freedom processing service through MotivateXR, then the CMS connects with 2F to start the process to generate a 3D model. Once ready, the 3D model 2F tool will use an API KEY to stored within the project's assets, making it available for use across other tools in the platform.

In the case of footage captured with a 2Freedom XVS Scanner, the process differs slightly. The scanner automatically uploads the footage to 2Freedom's servers, where the 3D model is generated. However, to make this model available within the MotivateXR platform, the user must manually download the 3D model from 2Freedom and then upload it to their project in MotivateXR.

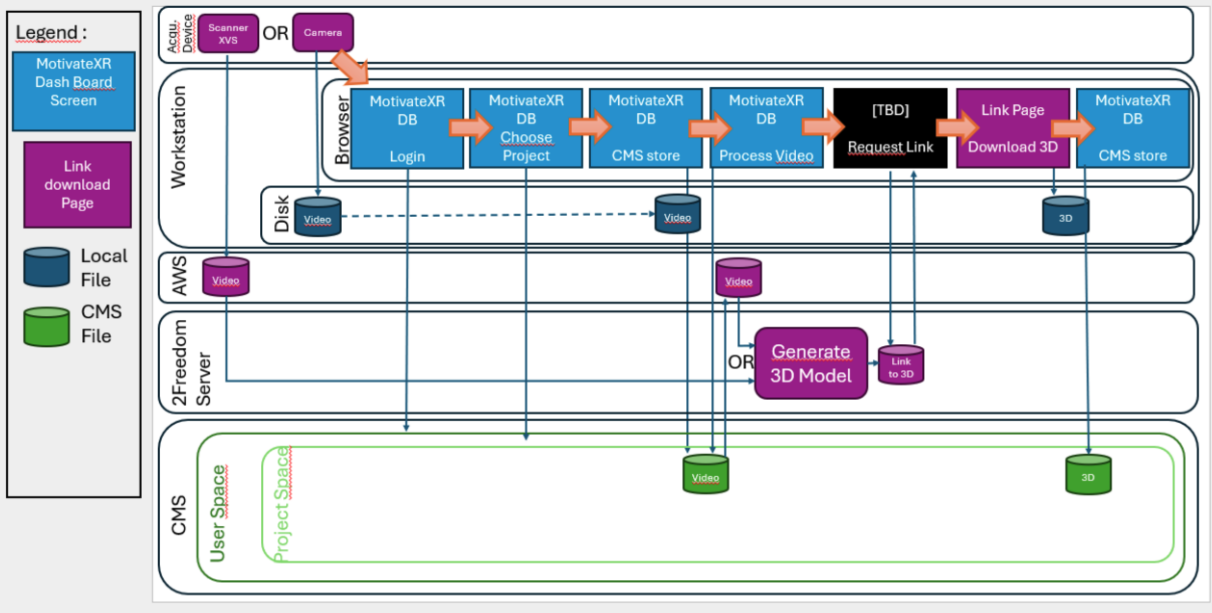

FIGURE 2 3D VIDEOGRAMMETRY SYSTEM FLOW

8.1.2 MIRA

Tool	MIRA
Description	MIRA is an authoring tool designed for creating and managing Digital Twins—virtual replicas of physical assets, processes, or systems
Integration Method	OAuth Authentication via an Identity Provider, API Key
Responsible	Maggioli

TABLE 9 MIRA INTEGRATION

The MIRA tool offers two complementary integration flows to manage and enhance digital twins within XR experiences.

In the first scenario, the user starts by logging into the MotivateXR web application using the standard Identity Provider (IdP). The same user should also have access to one organization in MIRA. After creating or selecting a project, the user can use the user interface of the MotivateXR web application to connect the MIRA digital twin that they have access to with the chosen MotivateXR project. Connecting a digital twin is technically implemented by storing an API key as part of the project's assets. This API key can be retrieved and used by other applications that belong to the MotivateXR ecosystem to access the digital twin data. More specifically, the XR authoring tools can use the API key to allow users to include Digital Twin's data as part of their immersive training and learning content during the authoring process (see Deliverable 5.1).

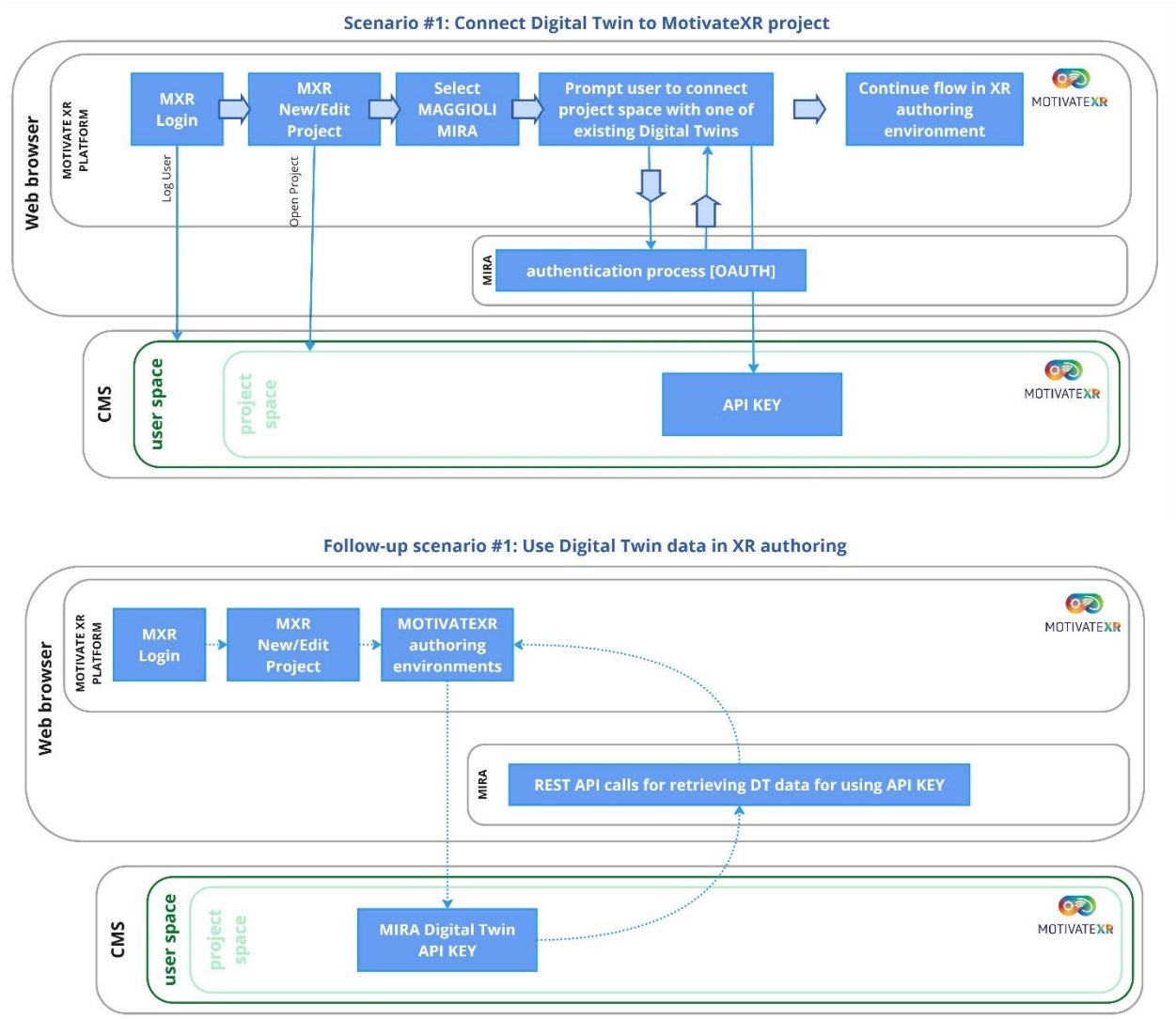

FIGURE 3 MIRA SYSTEM FLOW

In the second scenario, the user works directly from the MIRA web-based application. After logging into MIRA and selecting the desired organization, the user typically will monitor and manage its Digital Twins (assets, networks, etc.). The user should also have access to the MotivateXR platform and have a project space where 3D models are available. Such 3D models can have been uploaded by the user or created using other MotivateXR services, such as 2F's videogrammetry. The user can edit a desired asset in MIRA and assign to it the location (URL) of the respective 3D model, if available on the MotivateXR project space. This allows the user to associate 3D information with their digital twin counterpart as part of the MotivateXR ecosystem, thereby connecting the visual/spatial dimension with the digital twin dimension (see Deliverable 4.1).

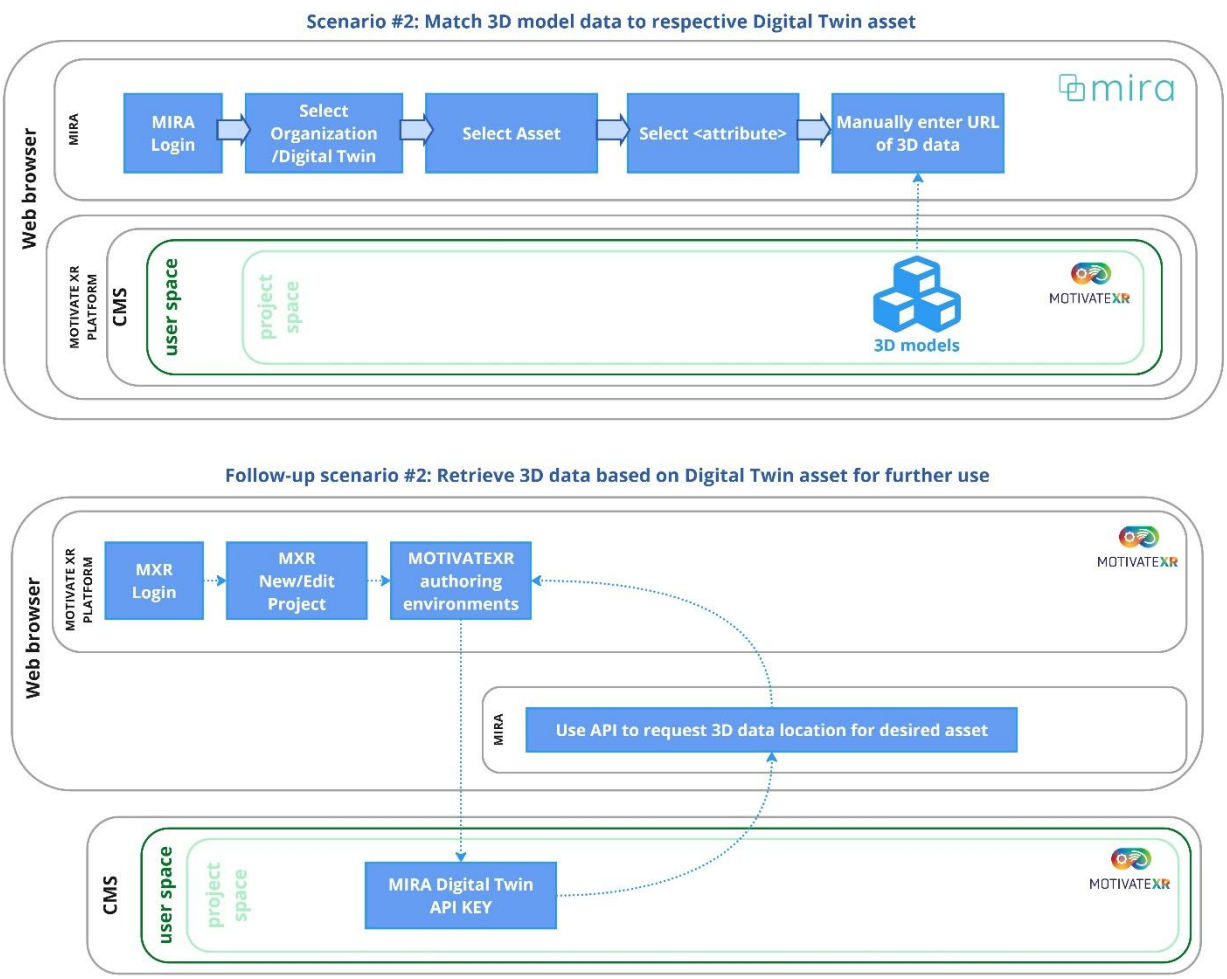

FIGURE 4 MIRA SYSTEM FLOW #2

With this bidirectional flow, digital twins and assets can be managed flexibly, whether the starting point is MotivateXR or MIRA.

8.2 XR AUTHORING TOOLS

Authoring Tools are the set of applications dedicated to creating and managing XR experiences within the platform. These tools allow users to design, edit, and organize all the necessary assets and content that form part of an XR experience, integrating them with the platform's services and delivery mechanisms.

8.2.1 KAYROX

Tool	KAYROX

Description	Authoring Tool for Creating XR Experiences
Integration Method	OAuth Authentication via Identity Provider
Responsible	TECNALIA

TABLE 10 KYROX INTEGRATION

The authoring flow for Kayrox starts with the user accessing the MotivateXR web dashboard, using the platform's Identity Provider (IdP) to authenticate. Once logged in, the user uploads or edit Assets, and creates or selects an existing project. From there, they can choose to start the authoring of the XR experience using Kayrox web app.
In parallel, the user could also access directly the Kayrox web app to edit an experience. The logging process would go through the Identity Provider (IdP) of the MotivateXR platform.

Within the Kayrox web app, users will be able to manage assets associated with the experience, create different steps of a sequential procedure, interactions, etc. They can save their progress at any time and retrieve it later for further editing.

Once the XR experience is finalized, the user can publish the experience, so that it will be available through the MotivateXR platform. Additionally, Kayrox will support importing content from MotivateXR services such as Document-to-3D and Video-to-3D conversion.

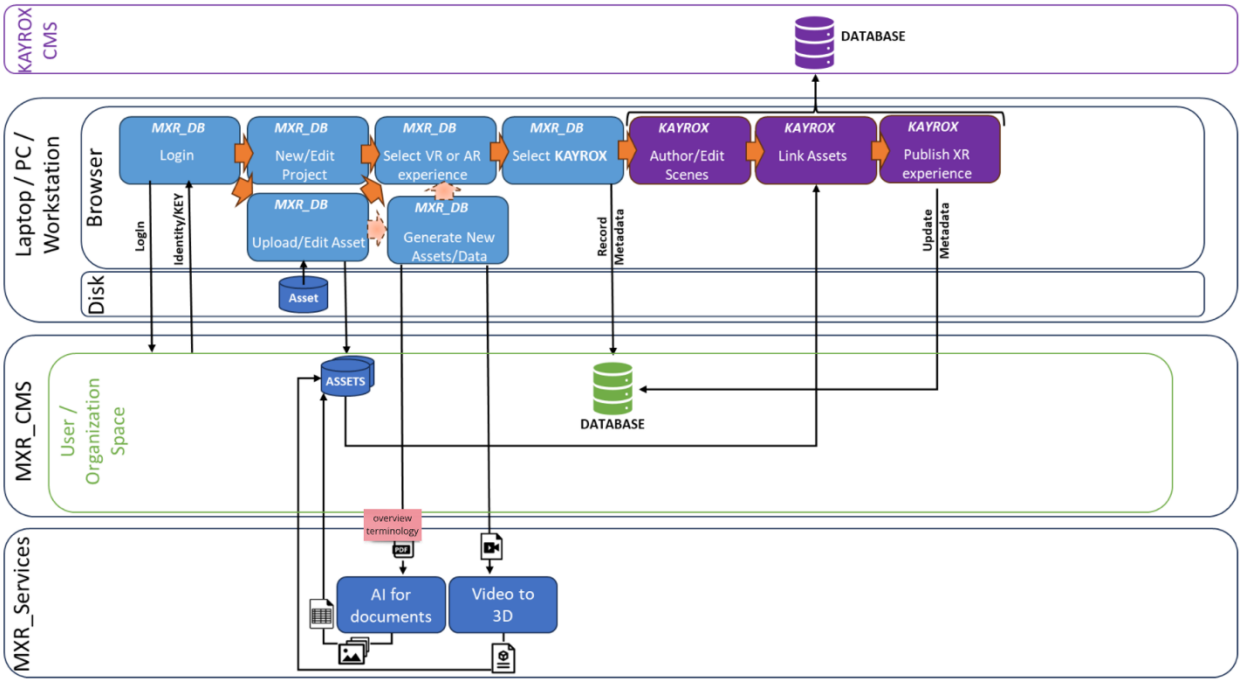

FIGURE 5 KAYROX AUTHORING TOOL FLOW

8.2.3 INSCAPE VTS EDITOR

Tool	Inscape Editor
Description	Authoring Platform for Interactive Storytelling
Integration Method	Dashboard
Responsible	CS

TABLE 11 INSCAPE VTS INTEGRATION

The integration flow for Inscape starts with the user accessing the MotivateXR web application, where they authenticate using the platform's Identity Provider (IdP). Once logged in, the user creates or selects an existing project, and from there, they can choose to create a new XR experience using Inscape. This action redirects the user to the Inscape landing page, where they are prompted to install the Inscape desktop application on their workstation if they haven't done so already.

Once Inscape is installed and launched, if the selected project in MotivateXR contains existing assets, such as FBX files or AI-generated assets from the Text AI Services (Semantic Processing Engine), the user can download these assets locally and import them into Inscape to be used in their XR experience. Inside Inscape, the user can then create or edit new XR content, working directly with the imported assets or adding new ones.

When the XR experience is completed, the final version can be uploaded back to the MotivateXR platform, making it available as part of the project for further use or distribution. Additionally, once published, the user will be able to download the packaged XR experience from MotivateXR if needed.

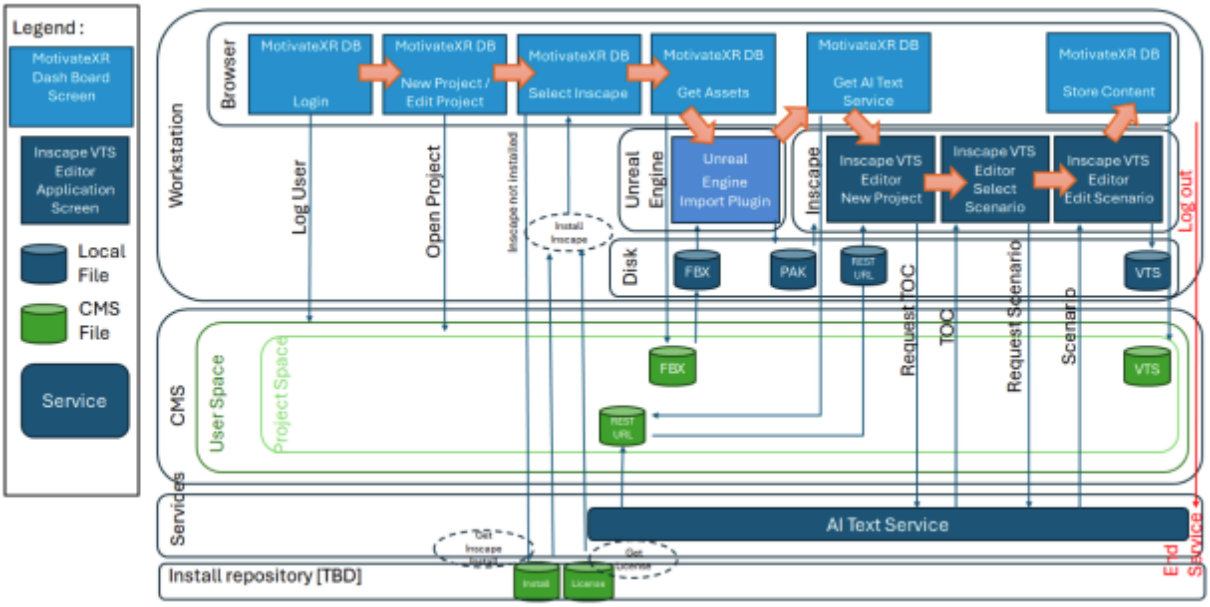

FIGURE 6 INSCAPE VTS EDITOR TOOL FLOW

8.2.5 NARRATIVE EDITOR

Tool	Narrative Editor
Description	Authoring tool designed develop story-driven, interactive experiences within immersive environments
Integration Method	Dashboard
Responsible	D3

Table 12 NARRATIVE EDITOR Integration

The Narrative Editor workflow begins with the user logging into the MotivateXR web application, authenticated via the platform's Identity Provider (IdP). After authentication, the user creates or selects a project within MotivateXR. If the project already contains FBX assets, these can be downloaded to the user's local environment for further use, or alternatively, users can retrieve FBX files directly from their local storage.

Other functionality involves uploading a PDF document to the MotivateXR platform and then trigger the AI Semantic Processing Engine. This service processes the document and automatically generates a JSON file containing structured information derived from the PDF.

The user can then import both the JSON file and the FBX assets into the Narrative Editor, where they can create or refine interactive XR scenarios based on this content. Once the scenario is

completed, it can be saved back to the MotivateXR Content Management System (CMS), making it available as part of the project for use within other MXR tools and experiences.

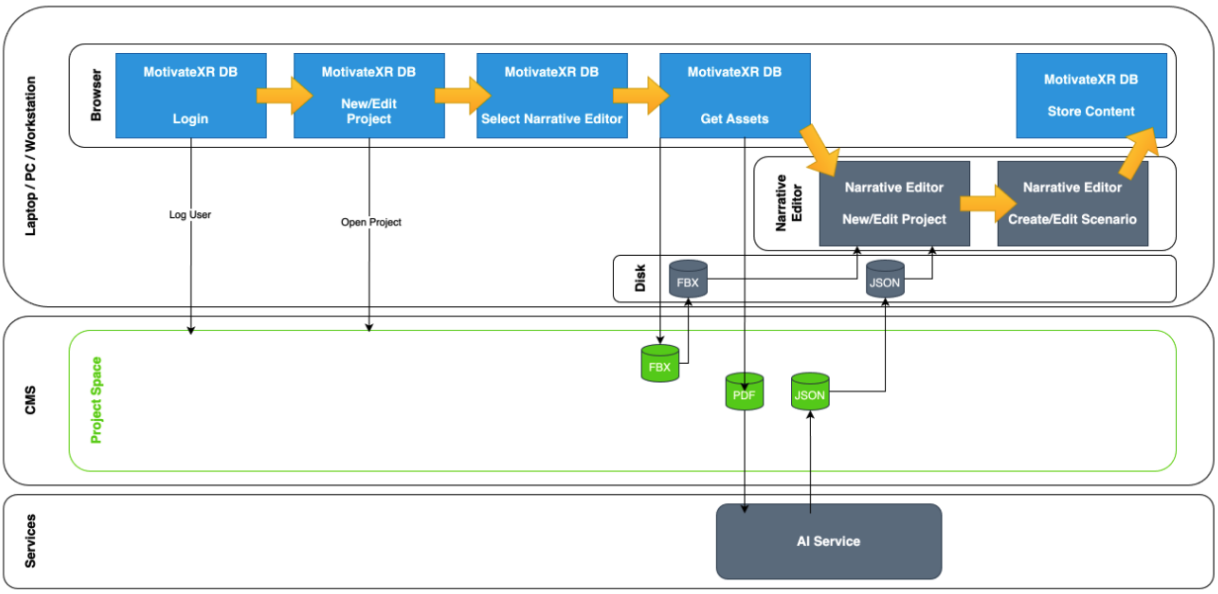

FIGURE 7 NARRATIVE EDITOR TOOL FLOW

8.3 AI SERVICES

The AI Services are a set of intelligent tools designed to enhance and transform assets, making them ready for use across authoring and experience tools within the MotivateXR platform.

8.3.1 SEMANTIC PROCESSING ENGINE (SPE)

Tool	Semantic Processing Engine
Description	SPE analyzes technical documentation, including text and illustrations, to extract structured content for use in authoring tools
Integration Method	API Key Authentication
Responsible	SOP, D3, UPM

TABLE 13 SPE INTEGRATION

The Semantic Processing Engine (SPE) is a service designed to process documents and enable intelligent querying of PDF content within the MotivateXR platform. It takes as input a PDF document and a thesaurus, allowing authoring tools to retrieve contextual information and generate dynamic responses based on the document's content.

The flow starts with the user logging into the MotivateXR web application through the Identity Provider (IdP), then selecting an existing project or creating a new one. Within the project, the user can upload or select a PDF document and select or create a thesaurus that will guide the semantic analysis. Once these elements are set, the user triggers the processing of the PDF via an API call to the SPE service.

The SPE then analyzes the PDF, extracts its content, and generates a knowledge graph stored in a Neo4j database instance, while also retrieving and storing associated images. After this process, the analyzed document is made available to authenticated users within the platform.

From that point on, authoring tools can send textual queries to the SPE, asking questions or requesting information based on the processed document. The SPE interprets these queries using the knowledge graph and the provided thesaurus, and returns structured responses in JSON format, ready to be used in XR content creation or as part of dynamic experiences.

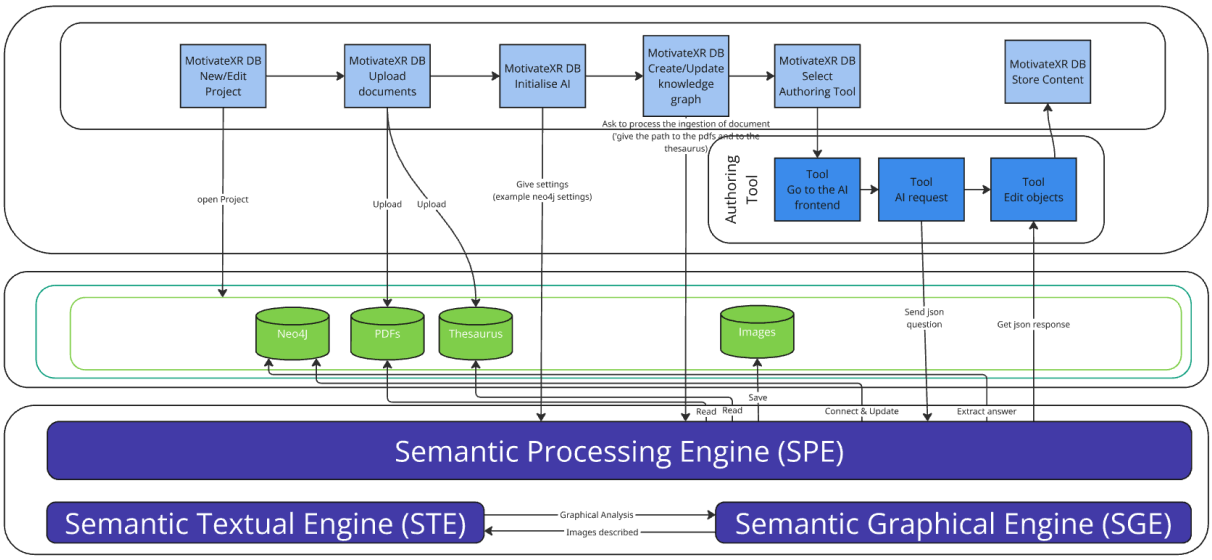

FIGURE 8 SPE TOOL FLOW

8.3.2 VISION MODULE

Tool	Vision Module
Description	Module to annotate videos and enhance XR experinces

Integration Method	API Key Authentication
Responsible	D3

TABLE 14 MIRA INTEGRATION

The Vision Module is an AI-powered service designed to process visual content in real-time and enhance it with AI-generated annotations, such as coordinate mappings and other relevant overlays. This service enables dynamic and interactive XR training experiences.

The flow begins when an authenticated experience tool (such as RTXR or other integrated XR applications) sends an image or video frame to the MotivateXR Content Management System (CMS). The CMS then triggers the Vision Module, which analyzes the frame and applies intelligent annotations based on the content.

Once processed, the annotated frame is returned to the CMS via API KEY, which return it to the experience tool and integrates it back into the live XR experience, allowing users to see enriched visual information in real-time.

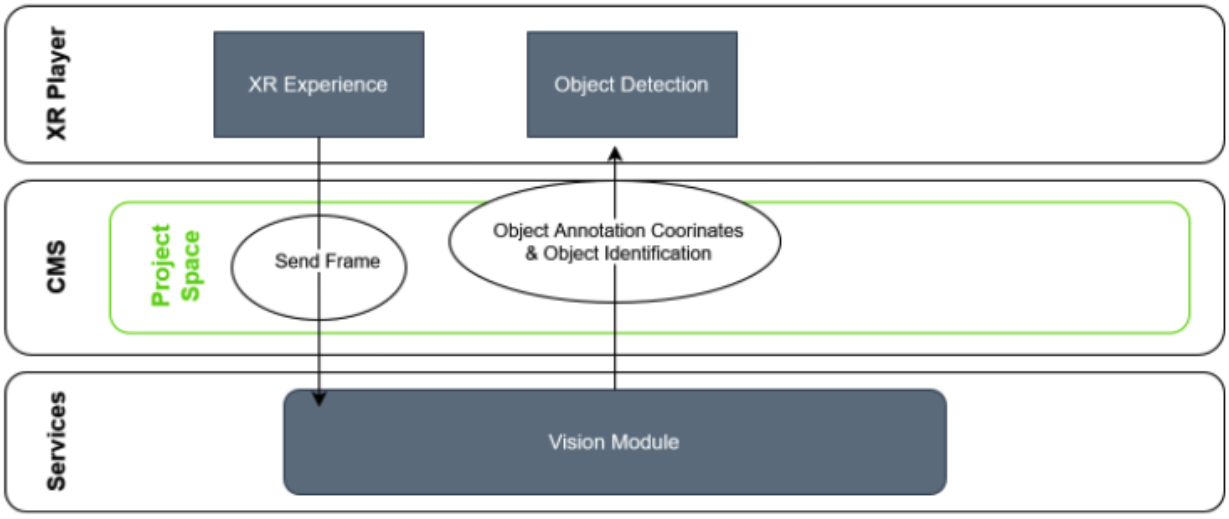

FIGURE 9 VISION MODILE TOOL FLOW

8.4 XR EXPERIENCING TOOLS

Experiencing Tools are the set of applications that allow users to interact with and visualize XR experiences created within the MotivateXR platform. These tools deliver immersive and interactive content, combining 3D assets, AI-enhanced data, and user-driven scenarios to provide training and

learning environments. They serve as the final step where all authored content and services come together to create XR experiences for end-users.

8.4.1 REMOTE TRAINING SYSTEM WITH AUGMENTED REALITY (RTXR)

Tool	RTRX
Description	Remote Training System with Augmented Reality
Integration Method	Dashboard
Responsible	D3

TABLE 15 MIRA INTEGRATION

The integration flow begins with the user accessing the MotivateXR web application, where they authenticate through OAuth using MotivateXR's Identity Provider (IdP). After successfully logging in, the user selects the desired project directly within the MotivateXR web app. Once the project is selected, the user chooses to launch the RTXR module, which opens the RTXR application tied to the selected project.

Inside RTXR, the user can browse and select an XR experience associated with the project. Upon selection, RTXR retrieves all the required assets for the experience — including FBX models, documents, and JSON configurations — from the MotivateXR platform.

As the user runs the XR experience on a display device, RTXR sends real-time video frames to the Content Management System (CMS) through a secured API using API-key authentication.

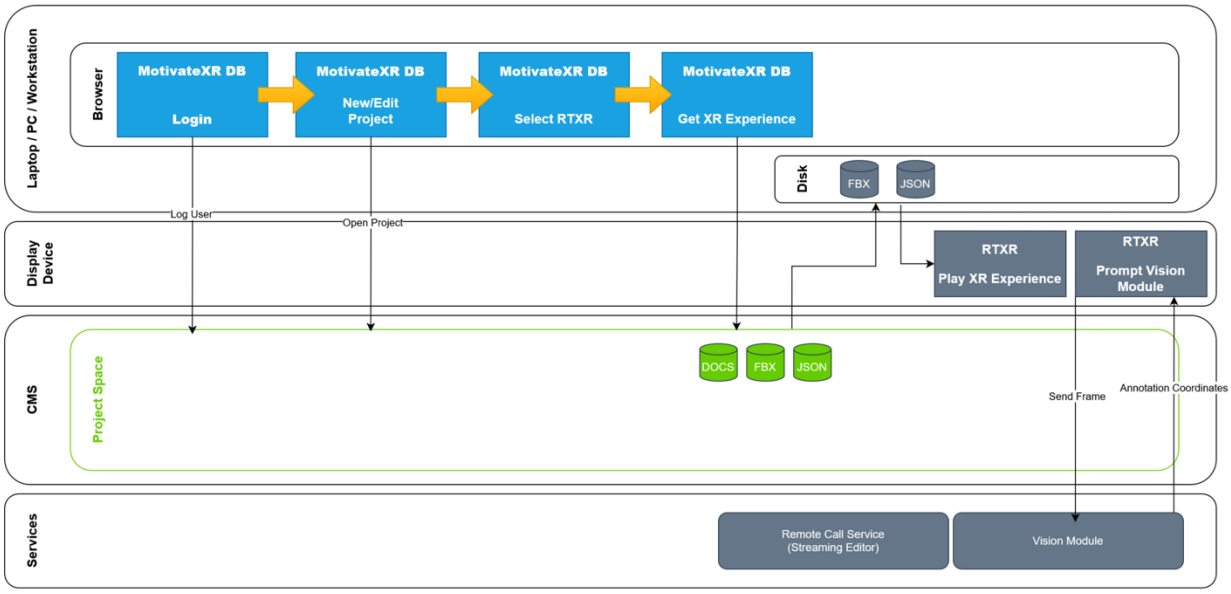

FIGURE 10 RTXR TOOL FLOW

8.4.2 INSCAPE VTS PLAYER

Tool	Inscape Player
Description	XR Experiences Player
Integration Method	Dashboard
Responsible	CS

TABLE 16 MIRA INTEGRATION

The Inscape VTS Player allows users to play XR experiences created and managed within the MotivateXR platform. The flow begins with the user logging into the MotivateXR web application, where they can browse available projects and experiences. After selecting the desired experience, the user chooses to play it using Inscape, which makes the experience available for download.

Once downloaded, the user imports the experience onto a display device equipped with the Inscape VTS Player, where it can be launched and played.

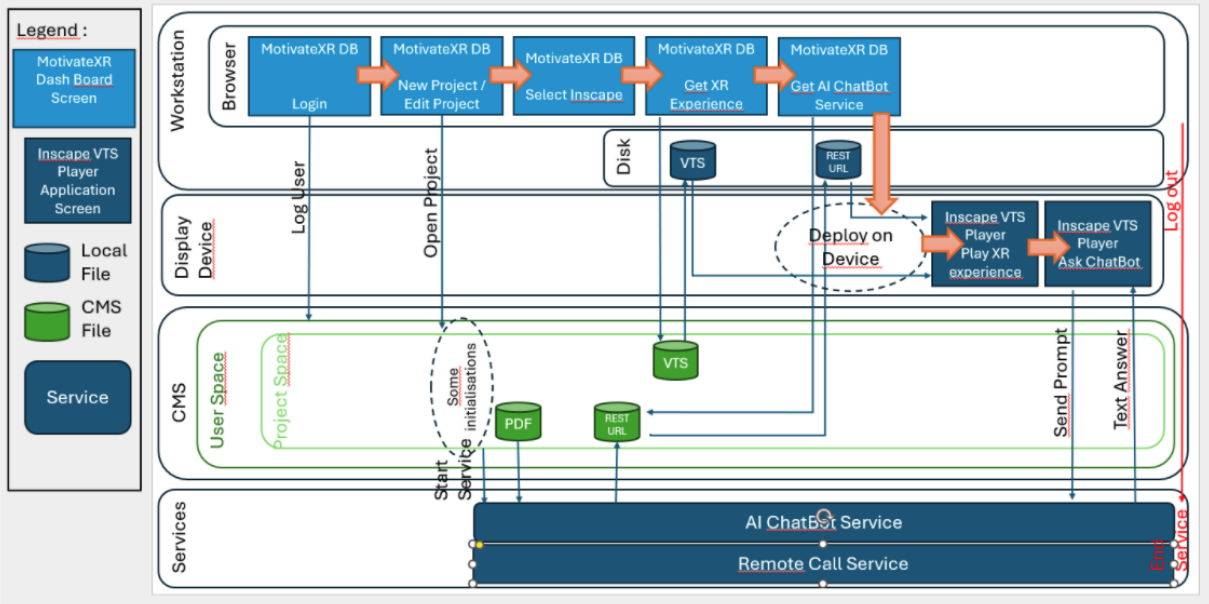

FIGURE 11 INSCAPE PLAYER TOOL FLOW

8.4.2 KAYROX PLAYER

Tool	KAYROX Player
Description	XR Experiences Player
Integration Method	OAuth Authentication via Identity Provider
Responsible	TECNALIA

TABLE 17 KAYROX PLAYER INTEGRATION

Kayrox Player provides two applications for consuming XR content: one player dedicated to AR/MR experiences and another focused on VR experiences. These players allow users to access and interact with XR content created within the MotivateXR platform.

- AR/MR native Player

This Kayrox app will be fully integrated with MotivateXR authentication service, allowing users to log in directly from the device. Upon authentication, the app retrieves the list of available experiences for the user via the MotivateXR CMS API, without the need to initiate the flow from the MotivateXR web platform.

Alternatively, users can also start from the MotivateXR web dashboard, where they browse projects, select an experience, and choose to launch it with a Kayrox device. In this case, the system generates a deep link (an additional method based on QR codes scan is being studied), that directly launches the Kayrox app and access the selected experience.

Once an experience has been selected to be experimented (in any of the 2 options explained), the app downloads to the device the necessary assets from MotivateXR CMS, along with needed metadata from Kayrox, and runs the XR experience.

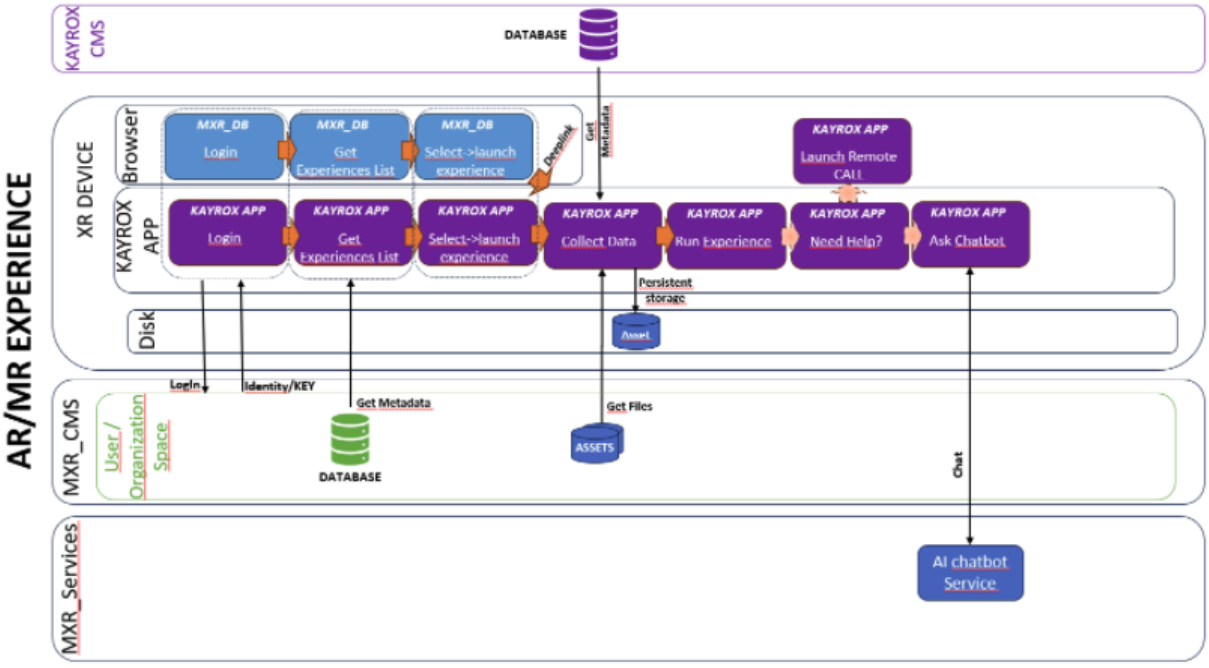

FIGURE 12 KAYROX AR/MR PLAYER FLOW

- VR web Player

The Kayrox VR web Player focuses exclusively on delivering VR experiences to users.

The experiencing flow starts with the user accessing the MotivateXR web dashboard. Once logged in, users can browse the different existing projects, select a published experience, and choose to experience it with Kayrox. This will launch the Kayrox web player directly. (As with the AR/MR experiences, direct access to the experience by scanning a QR from XR glasses is under consideration)
In parallel, the user could access directly from the Kayrox web player, to launch a published experience. The logging process would go through the Identity Provider of the MotivateXR platform.

Because the focus is on immersive VR content, the Kayrox VR Player does not offer the "assistance" features present in the AR/MR Player. Instead, the user's primary goal is to explore and interact with standalone VR experiences without real-time queries or additional contextual information.

Through this streamlined approach, the Kayrox VR Player ensures that users can quickly and efficiently immerse themselves in VR projects developed and managed within the MotivateXR platform.

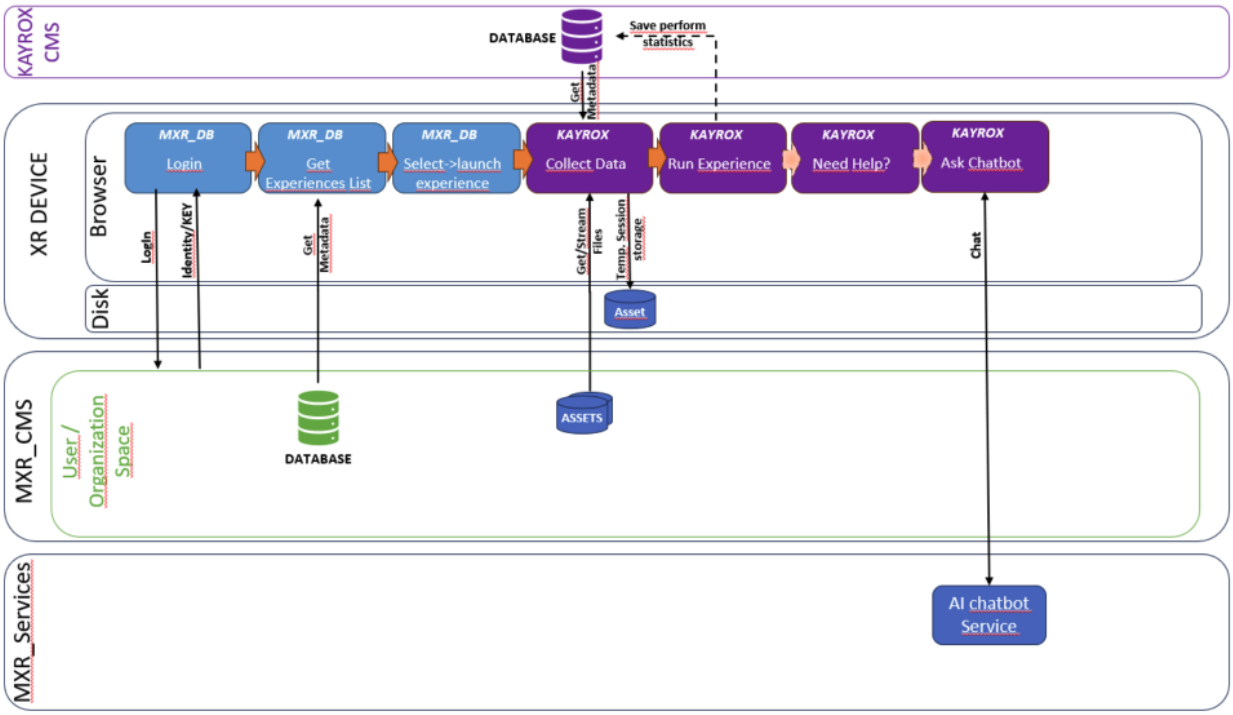

FIGURE 13 KAYROX VR PLAYER FLOW

8.5 WORKSHOP OUTCOMES

In conclusion, the integration methods described in this section represent a snapshot of how various tools can connect with the MotivateXR Platform, including the flow of data and the underlying authentication mechanisms. Each approach has been designed with all partners during the workshop to facilitate the integration while meeting diverse tool requirements. As we move toward full implementation, it is possible that details of these integrations may evolve in response to new features, user feedback, and the continued growth of MotivateXR's capabilities.

9 IMPLEMENTATION ROADMAP

The Gantt chart below shows the planned tasks and milestones for develop and deployment the components of the MotivateXR platform. By showing timing and overlaps, it demonstrates how infrastructure, backend services, and frontend interfaces will be developed in parallel, ensuring core functionality is established early and refined as the project progresses.

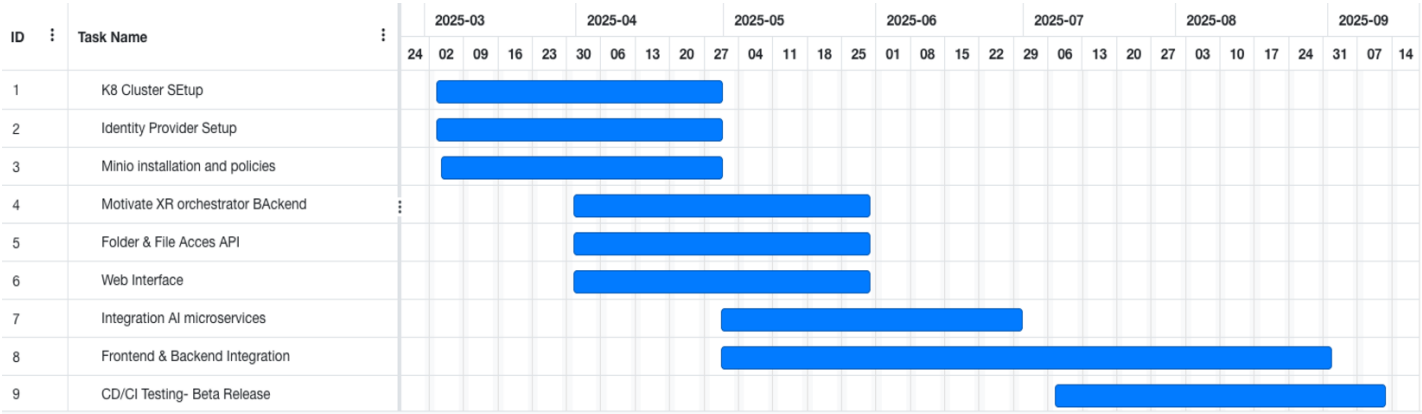

FIGURE 14 IMPLEMENTATION ROAD MAP

10 CONCLUSIONS

With this document, the continuous integration plan for the staging environment has been clearly defined, providing a structured workflow for building, testing, and deploying the components of the MOTIVATE XR ecosystem. It also presents a preliminary definition of how various tools and applications will interact with the platform, setting the basis for further refinement and allowing teams to move forward with a shared understanding of the integration approach.

The next deliverable will focus on detailing the integration plan for the production environment. The methods by which tools will authenticate, exchange data, and operate within MOTIVATE XR in a live setting will be defined. It is anticipated that some adjustments may be necessary as the project evolves and tools feedback and experiences will be considered.