MOTIVATE**XR**

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

D6.3 – MOTIVATE XR Integrated Release v1

30/09/2025

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

D6.3 - MOTIVATE XR INTEGRATED RELEASE V1

Work package	WP6
Task	T6.2
Due date	30/09/2025
Submission date	30/09/2025
Deliverable lead	MAG
Version	V1
Authors	Francisco Moreno (UPM), Olga Chatzifoti (MAG)
Reviewers	SOP, D3
Abstract	Announces the beta release of the integrated and verified MOTIVATE XR platform and offers an accompanying technical report and user manual. The platform Implements the design described in D6.1, integrates outputs of WP4 and WP5, and provides the technical environment for the first piloting session of WP7.
Keywords	Integration, CI, Cybersecure Platform

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	07.09.2025	First TOC generated	Olga Chatzifoti (MAG)
V0.2	12.09.2025	Section 2. Technical Description added	Francisco Moreno (UPM)
V0.3	15.09.2025	Section 4. Administrative capabilities added	Francisco Moreno (UPM)
V0.4	17.09.2025	Section 3. End-User Experience added	Olga Chatzifoti (MAG)
V0.5	18.09.2025	Section 1. Introduction, Executive summary and Section 5. Future Developments added	Olga Chatzifoti (MAG)
V0.6	19.09.2025	Text edits and formatting	Francisco Moreno (UPM), Olga Chatzifoti (MAG)
V0.7	26.09.2025	Internal review #1 complete	Paschalis Choropanitis (D3), Eleni Zisiou (D3)
V0.8	29.09.2025	Internal review #2 complete	Bruno Favresse (SOP), Lucas Colomines (SOP)
V0.9	30.09.2025	Review feedback incorporated	Francisco Moreno (UPM)
V1.0	30.09.2025	Submitted version	Nikos Achilleopoulos, Olga Chatzifoti, Alexandra Malouta (MAG), Francisco Moreno (UPM)

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
1.1	WARDEM SQUAD DATA-DRIVEN THINKING SL	WM	ES
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 GOSPODINSKI APARITI DOO	GOR	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 describes the beta release (v1) of the MOTIVATE XR platform, an integrated Extended Reality (XR) development and deployment environment, capable of creating, managing, and experiencing XR content across Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). It was developed during M7-M16 in accordance with the specifications set out in "D6.1 – Continuous integration plan and platform" (released on M6, with an extended version released on M10), and describes the work performed in "T6.1 Cybersecure Back-end and APIs development", "T6.2 Continuous Integration" and "T6.3 System Verification". Finally, the platform integrates outputs of WP4 and WP5, specifically "D4.1 Authoring tools beta" and "D5.1 Experiencing tools beta", while providing the technical environment for the first piloting session of WP7 (T7.2-7).

This deliverable combines theoretical explanations with practical demonstrations through screenshots and user interface examples, marking the document as both a technical report and a user guide for the beta release. The document is structured into five (5) main sections, each addressing different aspects of the MOTIVATE XR platform's beta release: Section 1 "Introduction" provides a high-level description of all aspects of the platform, while "Technical report" is covered in Section 2, "End User Experience" is covered in Sections 3, "Administrative capabilities" in Section 4 and "Future Development" in Section 5.

In more detail:

Section 1 - Introduction provides a comprehensive summary of the platform for quick onboarding.

Section 2 - Technical Report presents the complete system architecture, covering the six core modules (Frontend, API, Keycloak, PostgreSQL, MinIO, Redis) and their interactions. This section details the implementation approach for each component, from Next.js/React frontend development to FastAPI backend services and explains the Kubernetes deployment strategy.

Section 3 – End-User experience presents all key features of the platform, including User Management, File Management, relationship to XR Authoring and XR Experiencing applications, and the accompanying advanced processing services.

Section 4 - Administrative Tools provides guidance for platform administrators, covering both the Keycloak identity management console and MinIO storage management dashboard, including common administrative tasks and system monitoring capabilities.

Section 5 - Future Development outlines planned enhancements for the next release, including production readiness improvements and expanded integration capabilities.

TABLE OF CONTENTS

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
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

Platform Overview

The MOTIVATE XR platform is built on a modern, containerized microservices architecture deployed on Kubernetes infrastructure. The system integrates multiple specialized components to provide a complete XR development lifecycle:

- **Web-based User Interface**: A responsive Next.js/React frontend providing intuitive access to all platform capabilities
- **Secure Backend API**: FastAPI-based service gateway managing authentication, authorization, and business logic
- **Identity Management**: Keycloak-powered authentication with role-based access control (admin, editor, viewer)
- **File Storage System**: MinIO-based object storage with intelligent bucket organization and encryption
- **Temporary Session Management**: Redis implementation for secure QR-code based device authentication

Technical Architecture

The platform demonstrates modern cloud-native design principles with complete containerization and Kubernetes orchestration. The architecture separates concerns effectively:

- **Presentation Layer**: Next.js/React frontend with responsive design and role-based UI adaptation
- **Service Layer**: FastAPI backend providing RESTful interfaces and service orchestration
- **Data Layer**: PostgreSQL for identity management, MinIO for object storage, Redis for session management
- **Security Layer**: Keycloak providing centralized authentication, authorization, and user management

All components communicate through well-defined APIs with comprehensive authentication and authorization checks. The system supports both internal cluster communication and external service integration.

Deployment and Infrastructure

The current deployment runs on a four-node Kubernetes cluster with GPU support for AI processing services. External Network Attached Storage (NAS) provides persistent data storage through

Container Storage Interface (CSI) drivers, ensuring data durability across system updates and maintenance.

The continuous integration pipeline built on GitLab automates builds, testing, and deployments from the development branch to production, maintaining system reliability and enabling rapid iteration.

Key Features and Capabilities

The platform offers the following set of key features for the beta release:

User Management and Security

The platform implements a sophisticated multi-tenant security model where users can create and manage groups while maintaining strict access controls. Each user belongs to one group at a time, with roles (admin, editor, viewer) determining their permissions across all platform services. Authentication flows are streamlined through OpenID Connect integration, supporting both traditional web login and innovative QR-code based authentication for XR devices.

File Management and Organization

The storage architecture employs a three-tier bucket system:

- **Public buckets**: Shared read-only resources accessible to all authenticated users
- **Private buckets**: Personal storage spaces with full user control
- **Group buckets**: Collaborative workspaces with role-based permissions

All files are encrypted at rest, and access permissions are consistently enforced across the platform through integration between Keycloak roles and MinIO policies.

XR Content Creation and Authoring

The platform supports multiple XR authoring workflows by integrating with specialized tools:

- **KAYROX**: Web-based authoring with deep platform integration
- **INSCAPE**: Local VR development environment with hyperlink forwarding
- **NARRATIVE EDITOR**: Authoring tool for remote training XR applications

Projects are organized by immersion level (VR, AR, MR), with appropriate authoring tools automatically available based on project type. The platform provides seamless asset management and version control for XR development workflows.

Advanced Processing Services

Two key AI-powered services enhance content creation capabilities:

Video-to-3D Conversion Service: Employs videogrammetric processing to generate textured 3D models from video inputs, supporting multiple output formats (PLY, OBJ, FBX, GLB for meshes; LAS for point clouds) with configurable quality settings.

Semantic Processing Engine: Creates intelligent conversational agents from PDF documents, enabling natural language interaction with technical documentation and training materials through advanced semantic understanding.

XR Experience Delivery

While the web interface serves development and management functions, the platform's backend supports direct integration with XR devices and players. The QR-code authentication system eliminates the complexity of credential entry in virtual environments, enabling seamless access to platform resources during immersive experiences.

Administrative Capabilities

The platform provides comprehensive administrative tools through integrated dashboards:

- **Keycloak Admin Console**: Complete user, group, and role management with token configuration
- **MinIO Console**: Storage monitoring, bucket management, and policy configuration
- **System Monitoring**: Usage statistics, performance metrics, and security auditing

These tools enable efficient platform management while maintaining security and operational oversight.

The following sections present the details for each of the above-mentioned aspects.

2. TECHNICAL REPORT

The following sections provide a detailed description of the MOTIVATE XR platform for the beta release, including the architecture, implementation and deployment details.

2.1. MOTIVATE XR ARCHITECTURE

The platform is organized into several modules, each with a clear responsibility. Each module is containerized but together, they provide authentication, storage, and the core services that users interact with. The modules can be seen in Figure 1.

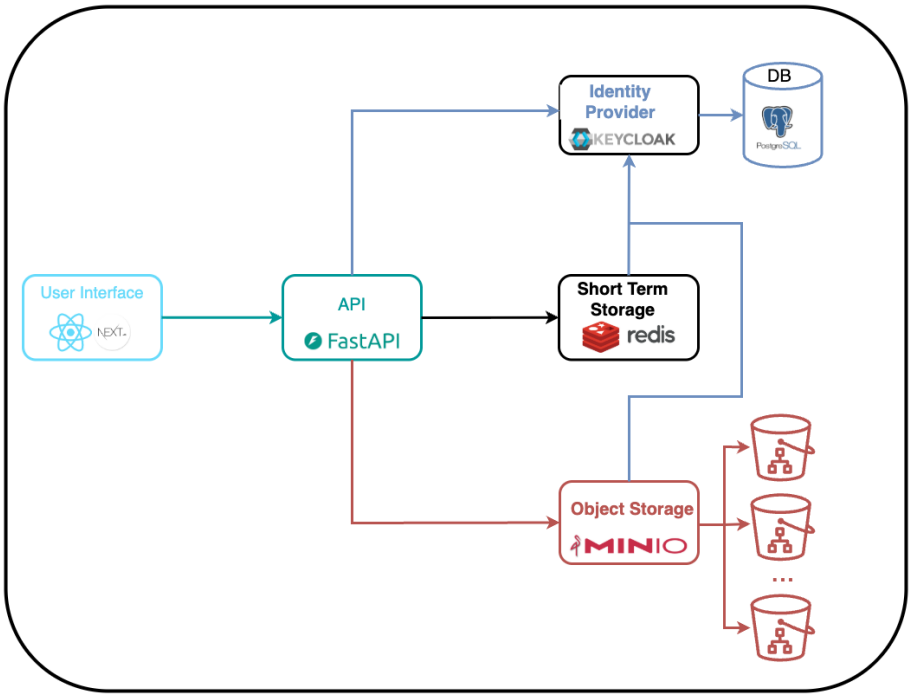

FIGURE 1 MOTIVATE XR MODULE ARCHITECTURE

2.1.1. FRONTEND (USER INTERFACE)

The frontend is built with Next.js[1] and React[2], providing a modern web-based interface for users. It allows them to log in, manage their files, and access the services of the platform. Authentication is integrated with Keycloak using the OpenID[3] Connect (OIDC) protocol. Once authenticated, the UI adapts to the user's role and group, showing or hiding features depending on their permissions.

[1] https://nextjs.org/
[2] https://react.dev/
[3] https://openid.net/

For example, an admin can configure settings, while an editor or viewer may only interact with files and services.

The frontend never communicates directly with storage or databases. Instead, all requests go through the API, ensuring that business logic and access control are consistently enforced.

2.1.2. API (SERVICE GATEWAY)

The API is implemented using FastAPI[4], chosen for its speed and ease of integration with modern authentication standards. It acts as the gateway between the frontend and the backend services. Developers can interact with the API to perform operations such as uploading or retrieving files, generating login links, or using application-specific services such as video to 3d and the semantic processing engine.

The API validates each request through Keycloak, checking the user's group and role before allowing access. It then coordinates with MinIO for file operations, Redis for temporary login tokens, and PostgreSQL (indirectly, through Keycloak) for identity data. This design makes the API the central piece of application logic, providing a single consistent entry point for all clients.

2.1.3. KEYCLOAK (AUTHENTICATION AND AUTHORIZATION)

Keycloak[5] is the identity and access management system of the platform. It runs with a single realm that controls authentication and authorization for all modules. Clients are defined for the API, the frontend, and external tools, all using OIDC as the integration protocol.

User management is built around groups and roles. Users can create groups, but they can only belong to one group at a time. Roles are global and include admin, editor, and viewer. These roles are applied across services to determine what actions a user can perform. For persistence, Keycloak relies on a standalone PostgreSQL database. Since the system is currently in beta, it runs as a single instance without replication or backup strategies.

2.1.4. POSTGRESQL (DATABASE LAYER)

PostgreSQL[6] is used exclusively as the database for Keycloak. It is deployed as a standalone instance within the Kubernetes cluster, with persistence handled by a CSI driver that connects to an external NAS. making data persistance independent to the cluster itself. No replication or backup is currently configured, as the platform is still in its early development phase.

2.1.5. MINIO (FILE STORAGE)

[4] https://fastapi.tiangolo.com/
[5] https://www.keycloak.org/
[6] https://www.postgresql.org/

MinIO[7] manages all file storage. It is deployed in standalone mode and uses policies tied to Keycloak groups and roles. Buckets are divided into three categories:

- Public buckets: available to all users with read-only access.
- Private buckets: accessible only to the bucket owner.
- Group buckets: shared among members of a Keycloak group, with permissions (view, edit, manage) mapped to their roles.

This design make access to files consistent with the authentication and authorization model, and MinIO enforces permissions directly at the storage layer.

2.1.6. REDIS (SHORT-TERM STORAGE FOR LOGIN LINKS)

Redis[8] is used as a short-term storage service that supports the login process on different devices. It stores temporary tokens that allow users to log in by scanning a QR code. These tokens expire after a short time and do not persist. Because Redis is only used for this ephemeral purpose, it runs without clustering or long-term storage. Its role is limited but important, ensuring a fast and reliable login experience.

2.2. IMPLEMENTATION

2.2.1. FRONTEND (NEXT.JS/REACT)

The front end of the platform is implemented with Next.js and React, providing a responsive web interface for end users. It is designed to give direct access to authentication, file management, and the set of services offered by the platform.

2.2.1.1. AUTHENTICATION & KEYCLOAK INTEGRATION

Authentication is managed with NextAuth.js, using Keycloak as the identity provider. The login flow is based on a redirect: when a user attempts to access a protected page, they are redirected to Keycloak, complete the login, and are then returned to the UI with an active session. Sessions are currently short-lived, without silent token refresh. User roles (admin, editor, viewer) and group membership are retrieved directly from the JSON Web Token (JWT) returned by Keycloak. These roles are used to control which UI components and actions are available.

2.2.1.2. UI FEATURES

The interface allows users to perform core actions on the platform:

[7] https://www.min.io/
[8] https://redis.io/

- Create VR/AR/XR repositories.
- Upload and download files.
- Create and manage groups, including user management for admins.
- Access and use platform services: video to 3D and semantic processing engine
- Open projects in external authoring and experiencing tools.

Role-based adaptation is applied in the UI: for example, administrators see user management options, while regular members only see the tools and projects they are allowed to work with.

2.2.1.3. COMMUNICATION WITH THE API

The frontend interacts with the backend exclusively via REST API calls. All requests are sent directly from the front end to the API, with authentication tokens attached. The API validates permissions with Keycloak before executing any action, ensuring that access control is enforced consistently.

2.2.1.4. BUILD & DEPLOYMENT

The front end is built as static compiled pages using Next.js' build process. This allows for fast load times and simple hosting. The compiled application is packaged and served with node as a Docker container, which is deployed in Kubernetes alongside the other services.

2.2.2. API (FASTAPI)

The backend API is implemented with FastAPI, providing a structured and high-performance gateway between the frontend and the platform's services. It exposes endpoints for authentication, group management, file operations, and external service integrations.

2.2.2.1. AUTHENTICATION & KEYCLOAK INTEGRATION

The API authenticates requests by validating Keycloak-issued JWT tokens. Instead of relying on static keys or Keycloak's introspection endpoint, the API dynamically retrieves the JWKS (JSON Web Key Set) from Keycloak's OIDC configuration. The appropriate public key is selected based on the kid in the token header, and the JWT is verified using RS256 The claims are then checked against the configured audience and issuer to check authenticity.

Authorization is applied at the endpoint level: each route enforces role-based access, returning a 403 Forbidden response when the user does not have sufficient permissions. The API consumes Keycloak tokens exclusively and does not issue its own.

2.2.2.2. FILE SERVICES (MINIO INTEGRATION)

The API exposes the full set of file operations — upload, download, delete, update, and folder management. In some cases, access is enforced directly through MinIO bucket policies, while in others the API itself checks user permissions against Keycloak roles before calling MinIO with an internal API key. This layered model allows flexibility: standard access flows rely on MinIO's policies, while sensitive operations are strictly validated by the API.

2.2.2.3. REDIS INTEGRATION (QR LOGIN)

Redis is integrated to support the temporary login process through QR codes. The API generates a short-lived token, stores it in Redis with a time-to-live, and encodes a URL containing the token into a QR code. Users can scan this code to log in, and the API resolves the QR token by retrieving it from Redis. Tokens expire automatically and are deleted after use, ensuring security.

2.2.2.4. EXTERNAL SERVICES

The API also acts as an orchestrator for additional processing services. Endpoints are exposed to start jobs such as video-to-3D conversion, PDF enhancement, and VR/AR/XR project creation. These services run asynchronously: one endpoint initiates the task, and another endpoint allows the client to query its status. Once complete, results (e.g., a processed 3D model) can be downloaded via the API.

2.2.2.5. ENDPOINT STRUCTURE

The API is organized into multiple routers for clarity and modularity:

- Groups: user and role management inside groups (/groups/...).
- Storage: repositories, file, and folder operations, tightly integrated with MinIO (/storage/...).
- Users: user search and QR login functionality (/users/...).
- Services: processing endpoints for 3D models and other content transformations (/services/...).

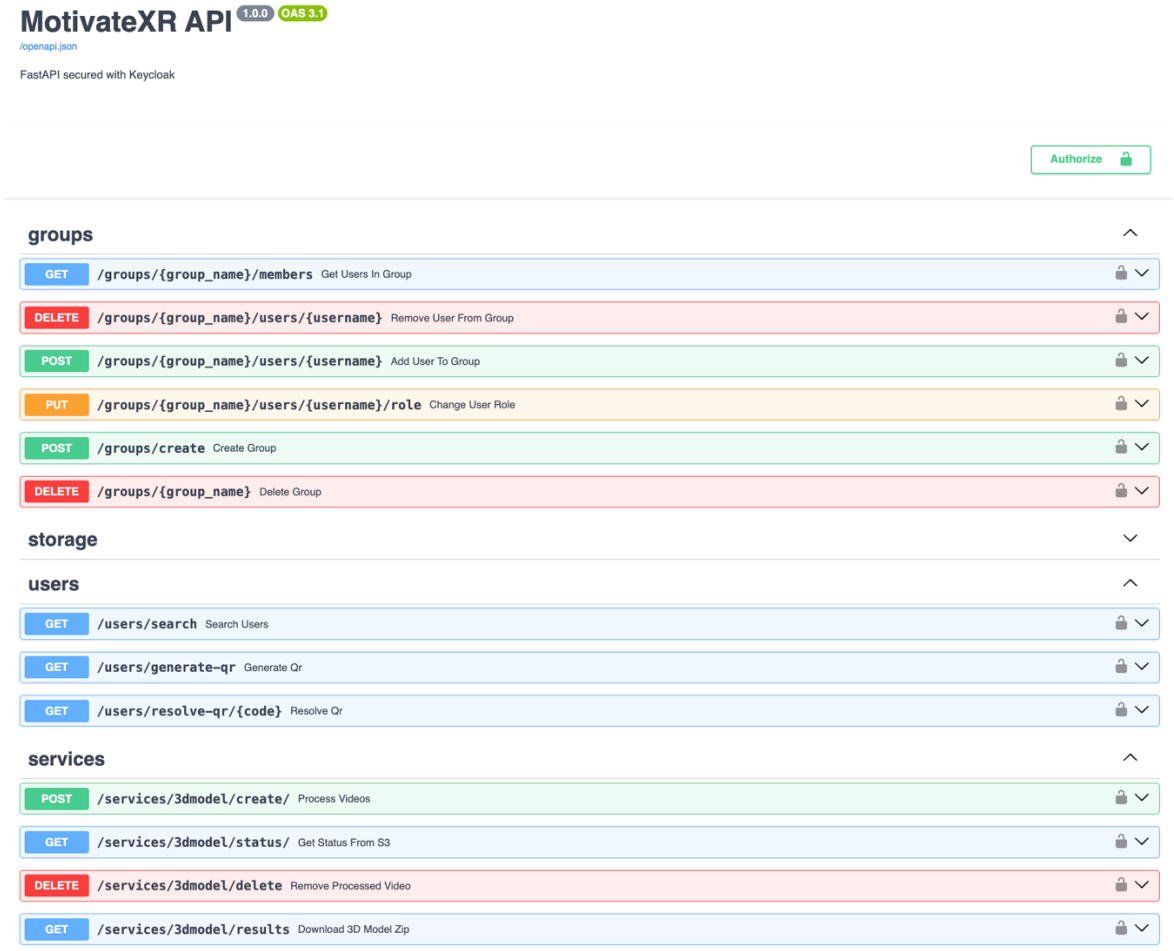

FIGURE 2 OPEN API DOCUMENTATION ENDPOINTS

2.2.2.6. DEPLOYMENT

The API is containerized using Docker. It is deployed as a single instance within the Kubernetes cluster, without a scaling strategy or load balancing. This is acceptable for early development stages but can be extended in the future with multiple replicas and Kubernetes' horizontal pod autoscaler.

2.2.3. KEYCLOAK CONFIGURATION

Keycloak provides the identity and access management layer for the platform. It is deployed in the Kubernetes cluster using a Helm chart, running as a single instance with a standalone PostgreSQL database for persistence. At this stage, Keycloak manages only internal accounts, with no integration to external identity providers.

2.2.3.1. REALM & CLIENTS

The system uses a single realm to manage all authentication and authorization needs. Within this realm, several clients are configured:

- A dedicated client for the frontend.
- A client for the API, which is responsible for securing backend operations.
- Confidential clients for external authoring and experiencing tools, which allow integration with the platform while preserving secure token handling.

All clients communicate with Keycloak through OpenID Connect (OIDC).

2.2.3.2. GROUPS & ROLES

Keycloak organizes users through a combination of groups and roles. Users may create their own groups, but they can only belong to one group at a time. The user who creates a group automatically becomes its owner and receives the admin role. Group owners cannot be removed from their role.

By default, new members of a group receive the viewer role, which grants them minimal permissions. Roles are defined globally as admin, editor, and viewer, but their effect is scoped to the group context. For example, an editor in one group may not have editor privileges in another. Making permissions consistent but isolated between groups.

2.2.3.3. SECURITY FEATURES

Authentication tokens issued by Keycloak have a lifetime of 30 minutes. They can be refreshed, and this behavior may be fine-tuned as the platform matures. At this stage, no special password policies or multi-factor authentication have been enabled, but Keycloak supports these features, and they can be added in future releases.

2.2.4. MINIO POLICIES

File storage in the platform is handled by MinIO, deployed in standalone mode with persistence backed by the external NAS through a CSI driver. MinIO is configured with encryption enabled by default, ensuring that all files are encrypted at rest.

2.2.4.1. BUCKETS & ORGANIZATION

The storage model is based on three top-level buckets: public, private, and groups.

- The public bucket is accessible to all authenticated users. Each user can create their own folders within this bucket to share content more openly across the platform.
- The private bucket is intended for personal files. Users create their own folders on demand, and only they can access their content.

- The groups bucket is automatically provisioned when a new group is created in Keycloak. A dedicated folder is created for the group, and its access is managed according to group membership and roles.

This structure provides a clear separation of concerns between shared, personal, and collaborative storage.

2.2.4.2. ACCESS POLICIES

Access control is enforced by integrating Keycloak with MinIO. Permissions are mapped directly from the roles defined in the group:

- Admin: full control over files and folders.
- Editor: read and write access.
- Viewer: read-only access.

In this way, the same role model defined in Keycloak is consistently applied to file operations. Even the public bucket requires the user to be logged in; truly anonymous access is not allowed.

2.2.4.3. SECURITY & ACCESS

All file operations go through the API, which acts as a proxy between the frontend and MinIO. This design makes sure that authorization checks are always applied, and it avoids exposing MinIO directly to end users. Downloads and uploads are validated against the user's roles before being passed to MinIO.

2.2.4.4. DEPLOYMENT

MinIO is deployed as a standalone server inside the Kubernetes cluster. Persistence is provided through the NAS, mounted with a CSI driver. This guarantees that files remain available across pod restarts and cluster upgrades.

2.2.5. REDIS SHORT-TERM TOKENS

Redis is integrated into the platform with a focused purpose: it provides short-term storage for QR-based login tokens. It runs as a standalone pod in Kubernetes without persistence since its role is limited to temporary data storage.

2.2.5.1. USAGE

The API uses Redis exclusively for the QR login feature. When a user requests a QR code, the API generates a short code and stores a corresponding JWT token in Redis. Each entry has a time-to-live of five minutes, after which it automatically expires, making login tokens remain valid only for a short window.

2.2.5.2. TOKEN HANDLING

Tokens are stored directly as JWTs. They are automatically removed once they expire and can also be deleted immediately after being used. This means that a QR code can only be resolved once to prevent reuse or replay attacks.

2.2.5.3. SECURITY

The keys in Redis are generated as short codes, derived from a six-character substring of a UUID. This makes them unique and difficult to guess. Short validity window and one-time use add an effective security layer for QR login.

2.2.5.4. DEPLOYMENT

Redis is deployed as a single pod inside the Kubernetes cluster. Since its data is ephemeral by design, no persistence or backup is configured.

2.2.6. POSTGRESQL PERSISTENCE

PostgreSQL serves as the database backend for Keycloak and is not used by any other service on the platform. It is deployed as a standalone instance within the Kubernetes cluster, without replication or clustering. This setup is sufficient for the current beta release, where high availability and failover are not yet required.

2.2.6.1. STORAGE

Data persistence is provided through a CSI driver that mounts an external NAS into the cluster. A dedicated Persistent Volume Claim (PVC) is bound to the PostgreSQL pod, ensuring that all identity and access data managed by Keycloak remains available across restarts or redeployments.

2.2.6.2. CONFIGURATION

The database is initialized and managed through the Keycloak Helm chart, which handles schema setup at startup. Since this environment is focused on development and early testing, no replication, backup, or point-in-time recovery features have been configured yet. These options remain available for future production deployments.

2.2.6.3. SECURITY

PostgreSQL runs as an internal service, with access restricted to the cluster. Connection credentials are managed as Kubernetes secrets, ensuring that database passwords are not exposed in deployment manifests.

2.3. DEPLOYMENT

2.3.1. KUBERNETES CLUSTER OVERVIEW

The platform runs on a Kubernetes cluster managed with Rancher, deployed on the UPM facilities. The cluster currently consists of four nodes, one of which is equipped with a GPU to support services that require hardware acceleration like the semantic processing engine.

All workloads are deployed under a single namespace, motivatexr, which simplifies management during the beta phase. This namespace contains the frontend, API, Keycloak, PostgreSQL, MinIO, and Redis, as well as the services that support content processing.

Since the environment is focused on early development, resource allocation is configured in a lightweight mode. No strict CPU or memory limits are enforced yet, leaving room for flexibility while the system evolves. Scaling strategies and resource optimization are expected to be introduced in later stages as usage increases.

2.3.2. STORAGE LAYER

The platform relies on an external NAS system for persistent storage, integrated with Kubernetes through a CSI driver. This provides stable and reliable volumes that remain available across pod restarts and redeployments.

2.3.2.1. POSTGRESQL PERSISTENCE

The PostgreSQL instance used by Keycloak is bound to a dedicated Persistent Volume Claim (PVC) connected to the NAS.

2.3.2.2. MINIO PERSISTENCE

MinIO also uses the NAS for persistence, storing its three primary buckets — public, private, and groups. Each bucket is hosted on the NAS, with encryption enabled by default.

2.3.2.3. REDIS STORAGE

Redis is deployed without persistence. Since its role is limited to short-term QR token storage, it uses ephemeral storage only.

2.3.3. SERVICE DEPLOYMENT

All core components of the platform are deployed as Kubernetes workloads inside the MOTIVATE XR namespace. Each service runs as a containerized application, with persistence configured through the external NAS where required.

2.3.3.1. KEYCLOAK

Keycloak is deployed using a Helm chart, configured as a single replica since high availability is not yet required. It is exposed through a Kubernetes Ingress, allowing both the API and frontend to redirect authentication flows to it.

2.3.3.2. POSTGRESQL

PostgreSQL is also deployed via a Helm chart, running as a StatefulSet. It is bound to a Persistent Volume Claim (PVC) that connects to the NAS through the CSI driver, ensuring that all identity and access management data remains durable.

2.3.3.3. MINIO

MinIO is deployed using custom Kubernetes manifests. It runs as a standalone instance, with a PVC bound to the NAS to provide persistence for its buckets (public, private, and groups). This setup guarantees durability of files while keeping the deployment lightweight.

2.3.3.4. REDIS

Redis is deployed through a Helm chart as a standalone pod. Since it is only used for short-lived QR login tokens, it runs without persistence and does not require a PVC. Its data resides entirely in memory and expires automatically.

2.3.3.5. FRONTEND (NEXT.JS/REACT)

The front-end application is built into a Node.js container that runs the Next.js runtime. It is deployed as a Kubernetes Deployment and exposed through an Ingress, allowing end users to access the web interface directly.

2.3.3.6. API (FASTAPI)

The backend API is packaged as a container running Uvicorn. It is deployed as a Deployment in Kubernetes with a single replica for now. Like the frontend, it is exposed externally through an Ingress, providing a REST interface to the UI and other clients.

2.3.4. NETWORKING AND ACCESS

The platform relies on a NGINX ingress controller to manage external access to its services. All ingress traffic passes through this controller, which routes requests to the appropriate backend service.

2.3.4.1. EXTERNAL ACCESS

The following services are exposed outside the cluster:

- Frontend: the main entry point for users, serving the Next.js/React UI.
- API: providing REST endpoints for the frontend and external clients.
- Keycloak: available for authentication, with direct login access restricted to administrators.
- MinIO: exposed for direct access with Keycloak credentials, though in future releases this may be hidden behind the API for tighter control.

All these services are served over HTTPS, ensuring encrypted communication. Access is additionally protected by firewalls at the infrastructure level, though there are no VPN restrictions at this stage.

2.3.4.2. INTERNAL COMMUNICATION

Inside the cluster, services communicate through ClusterIP services. The primary flows are:

- The API validates tokens against Keycloak.
- The API interacts with MinIO for file operations.
- The API uses Redis for short-lived QR login tokens.
- Keycloak persists its data to PostgreSQL.

Currently, network communication between pods is open; no Kubernetes network policies are enforced. This simplifies development, though it may be tightened in future iterations.

2.3.4.3. AUTHENTICATION FLOW

The typical authentication sequence starts when a user accesses the Frontend through the ingress. Login requests are redirected to Keycloak, and once completed, the user is returned to the UI with a token. The front-end then calls the API with this token, which validates it with Keycloak before allowing access to MinIO or other protected services. Administrators may also log in directly to Keycloak, and MinIO can be accessed directly with Keycloak credentials.

2.3.5. CONFIGURATION MANAGEMENT

Configuration across the platform is managed through a combination of Kubernetes Secrets and ConfigMaps, depending on the sensitivity of the data.

2.3.5.1. SECRETS

Sensitive values, such as database credentials, API keys, and Keycloak admin passwords, are stored as Kubernetes Secrets. These are mounted as environment variables into the respective pods at runtime. Examples include:

- PostgreSQL connection details for Keycloak.
- MinIO access and secret keys.
- Keycloak admin credentials.
- JWT and OIDC-related secrets for the API and frontend.

2.3.5.2. CONFIGMAPS

Non-sensitive configuration values are provided through ConfigMaps, such as:

- API endpoint URLs for internal communication.
- Service-specific environment variables (e.g., default token lifetimes).
- UI build settings for the front-end.

2.3.5.3. HELM INTEGRATION

For components deployed via Helm charts (Keycloak, PostgreSQL, Redis), configuration is encapsulated in Helm values files. This centralizes control and reproducibility of deployments. Custom values are provided for storage bindings, ingress settings, and runtime parameters.

2.3.6. DEPLOYMENT WORKFLOW

The platform uses a GitLab CI/CD pipeline to automate builds and deploy the UI and the API, Helm charts and manifests are updated manually. Every time changes are merged from the dev branch into main, the pipeline is triggered.

2.3.6.1. BUILD AND TEST

A self-hosted GitLab runner is installed on the UPM premises to execute the pipeline. The runner builds container images for the frontend, API, and supporting services using their respective Docker files. Before proceeding with deployment, the pipeline runs a set of unit tests. Only if these tests pass, does the process continue.

2.3.6.2. DEPLOYMENT

Once validated, the pipeline pushes the images to the container registry and applies the updated configurations to the Kubernetes cluster.

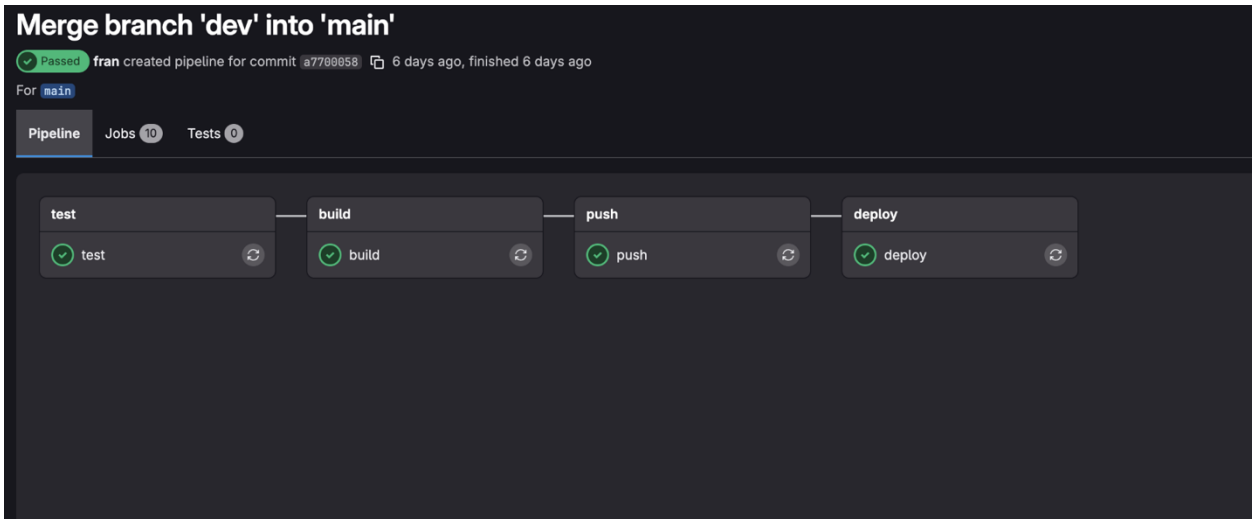

FIGURE 3 RESULTS OF THE CI PIPELINE

3. END-USER EXPERIENCE

3.1. USER LOG-IN

The user onboarding process starts with an email from the platform provider to the new user's email account with a temporary username and password. A user can modify their username and password on their first log in attempt.

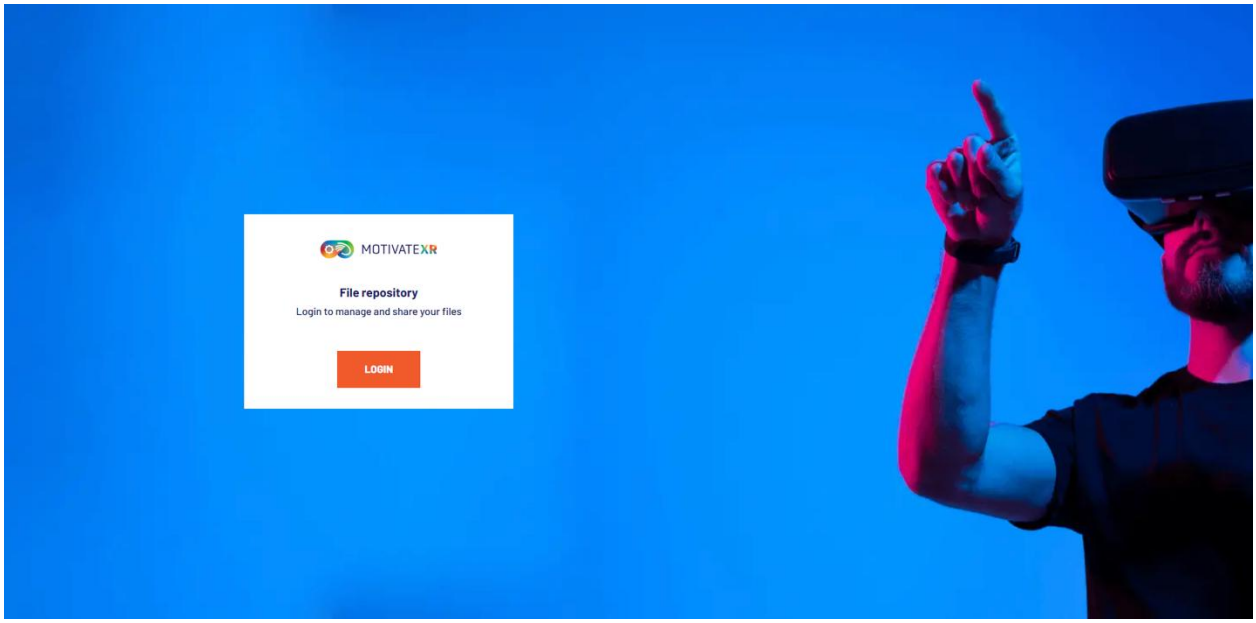

FIGURE 4 USER LOG-IN VIEW

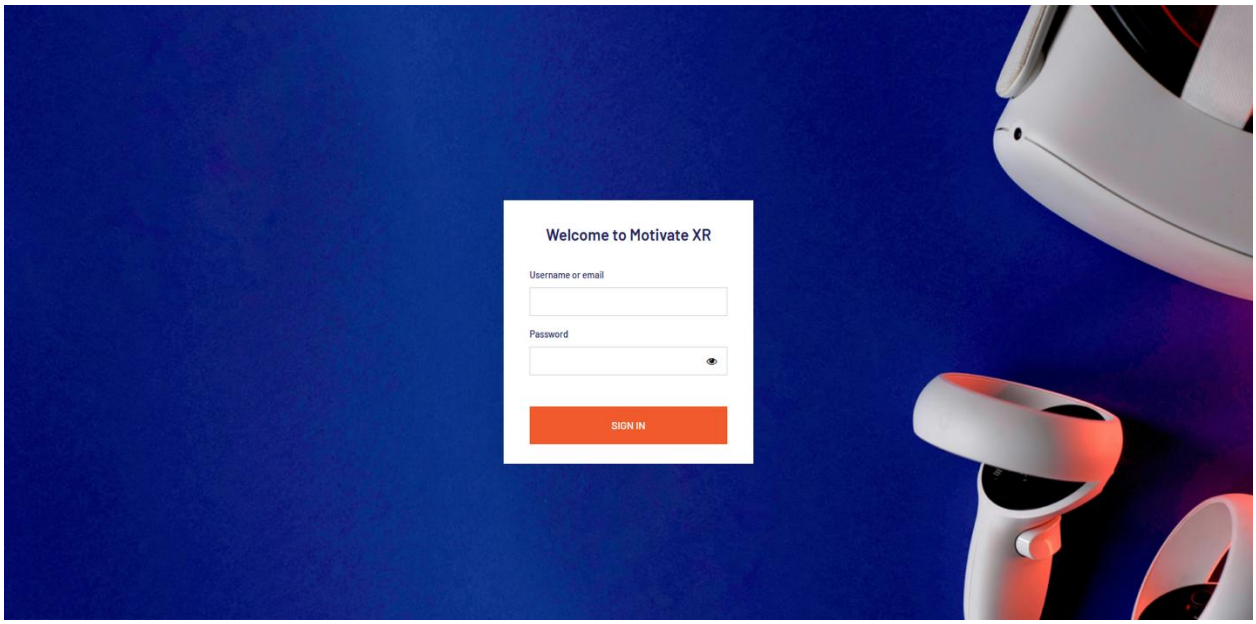

FIGURE 5 USERNAME AND PASSWORD

In addition, logged-in users can edit their user account details at any point by clicking on the user icon on the top right and selecting "User Settings" from the drop-down menu.

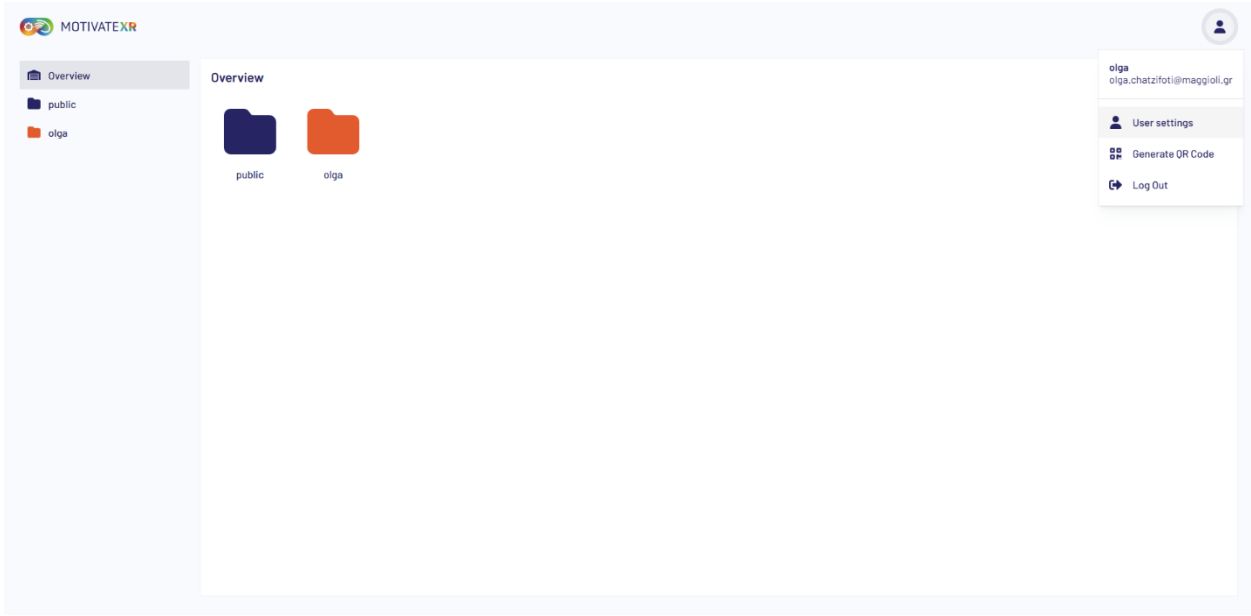

FIGURE 6 FOLDER CATEGORIES

3.2. FILE MANAGEMENT

Logged-in users can have access to two folders: the public folder, where users can preview folders and files and download (but not upload) files, and a private folder (named after the user's declared

username), where the user can create project folders. Within their private folder, users can create new projects using the respective UI element and select between “VR project”, “AR project” and "MR project”. Upon selecting a project type, the user can either start with a blank folder, or select one of the available tools to create a folder with a project template.

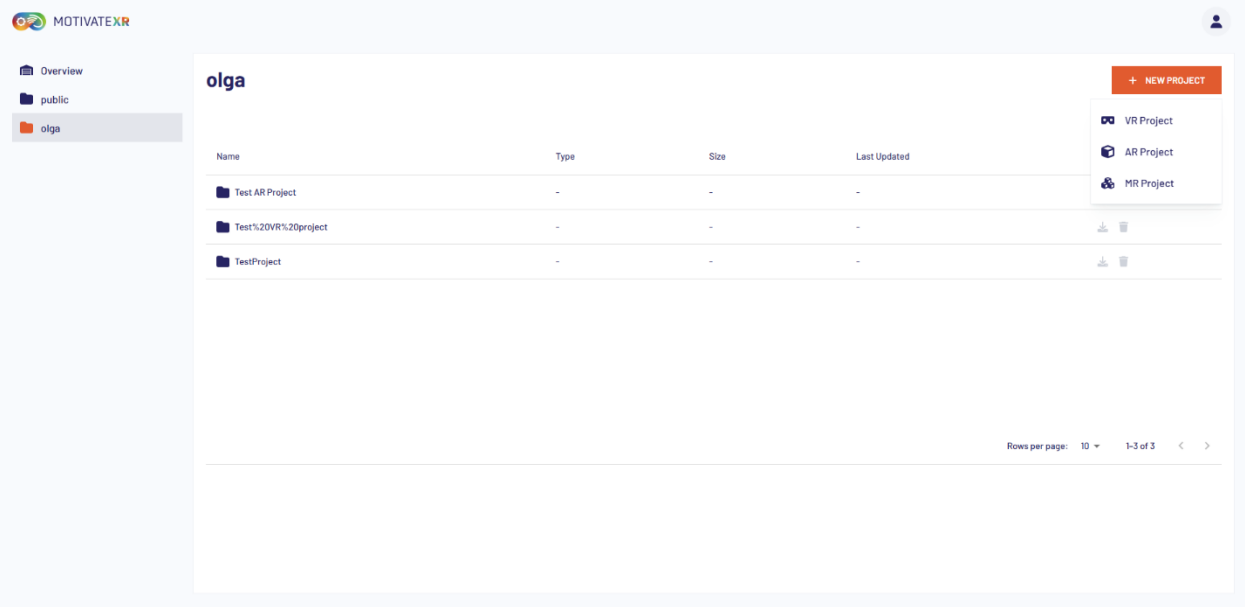

FIGURE 7 PROJECT CATEGORIES

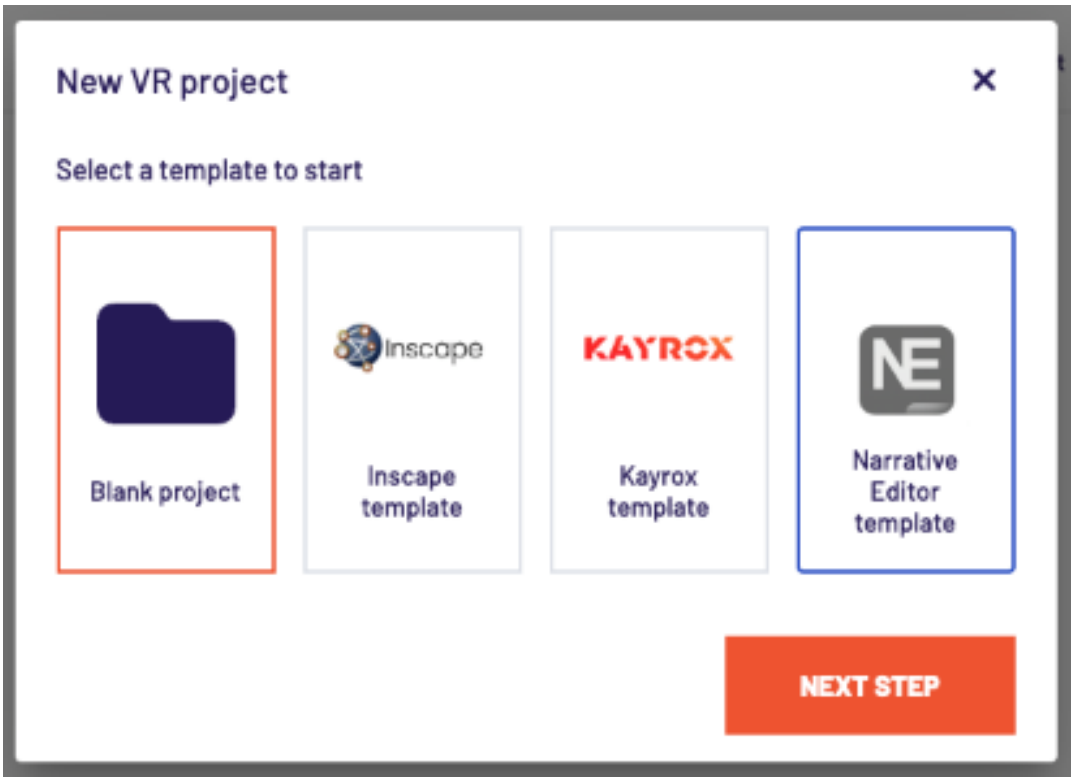

FIGURE 8 NEW PROJECT TEMPLATES

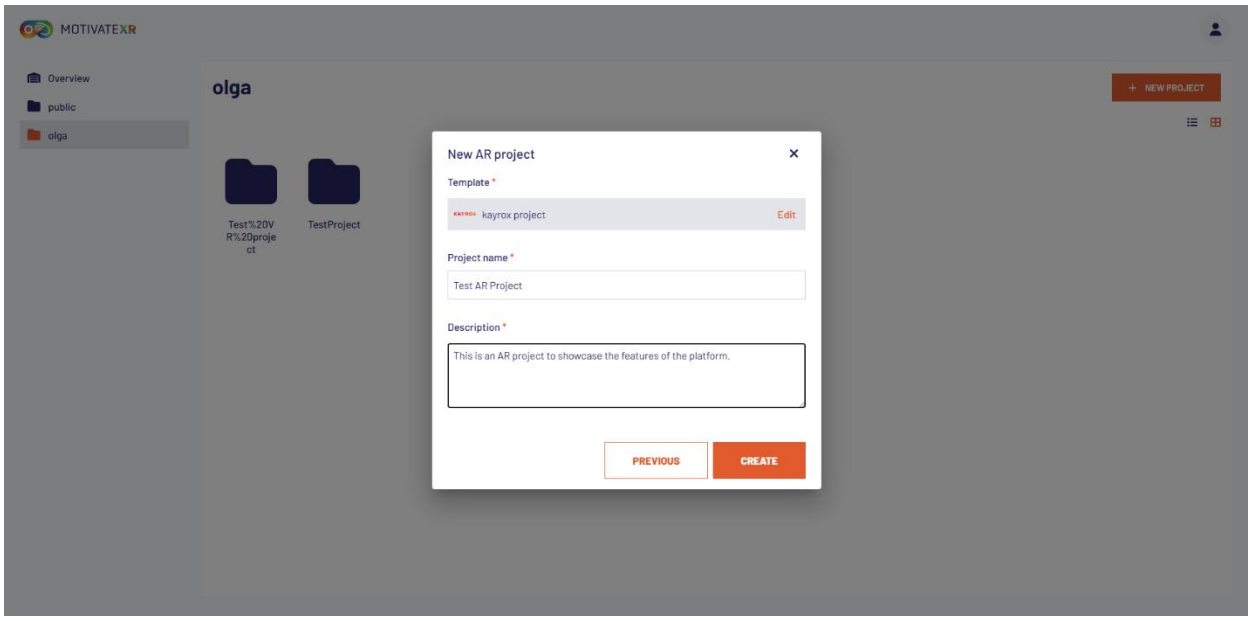

FIGURE 9 NEW FOLDER CREATION

The "New project" creation process requires a Project Name and Description and leads to the creation of a same-named project, within which users can upload their data.

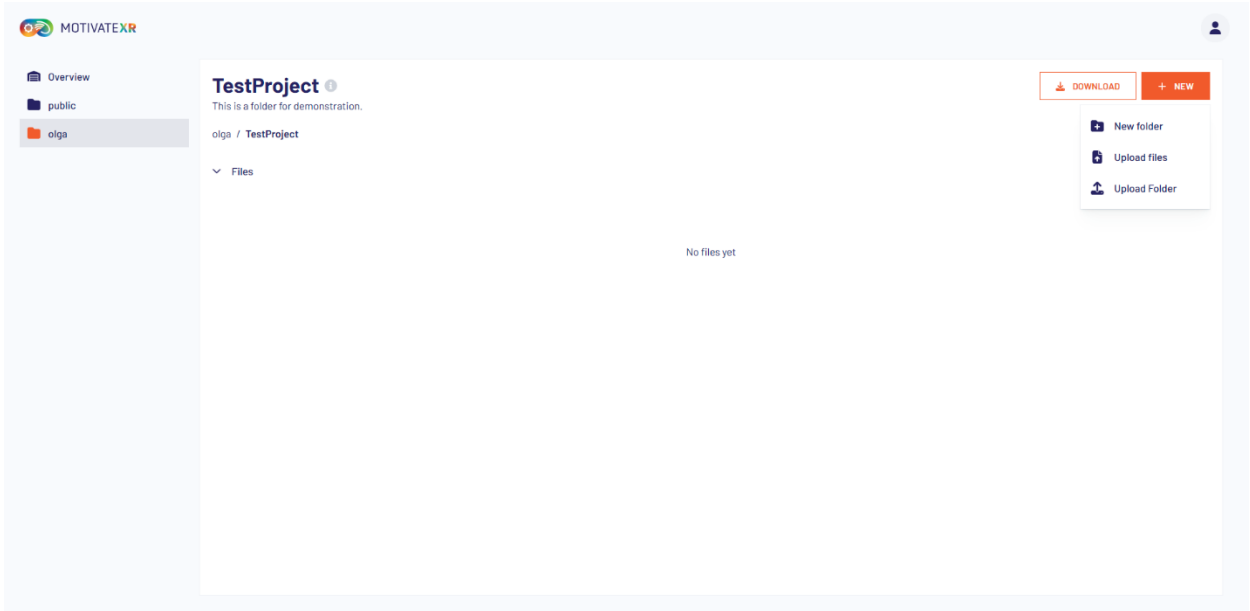

FIGURE 10 FILE AND FOLDER MANAGEMENT

Within the created folder, the user can upload their data either as individual files or as folders and can create further sub-folders to organize their data.

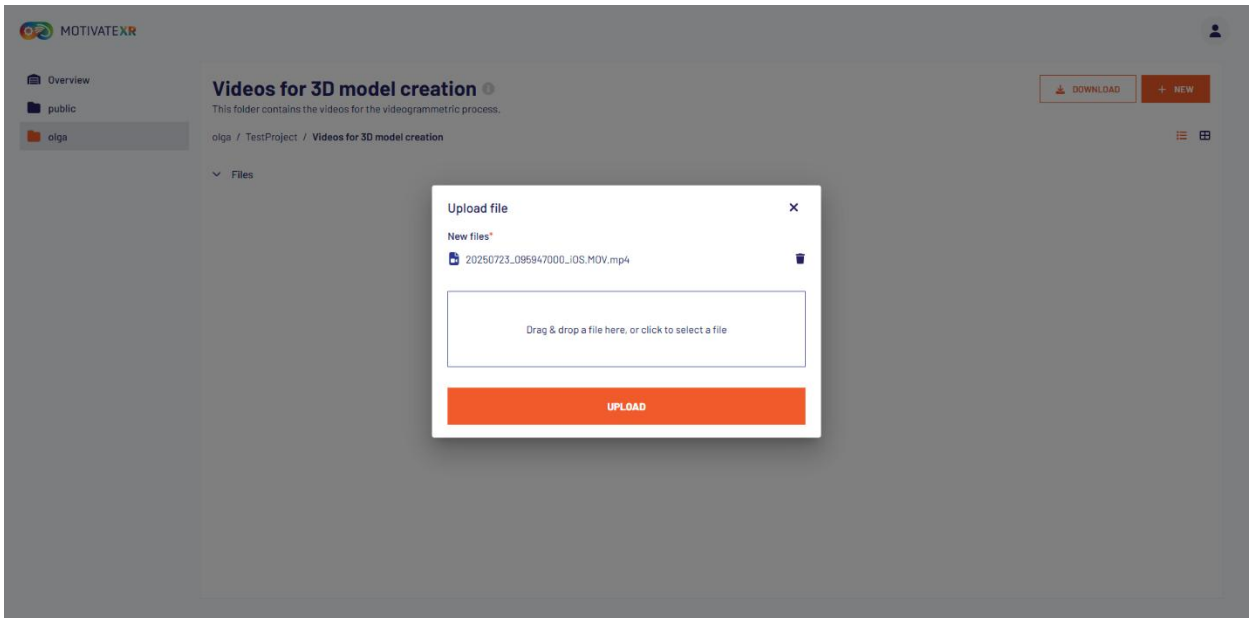

FIGURE 11 FILE UPLOAD PROCESS

Uploaded image files, videos and 3D objects can be previewed within the platform.

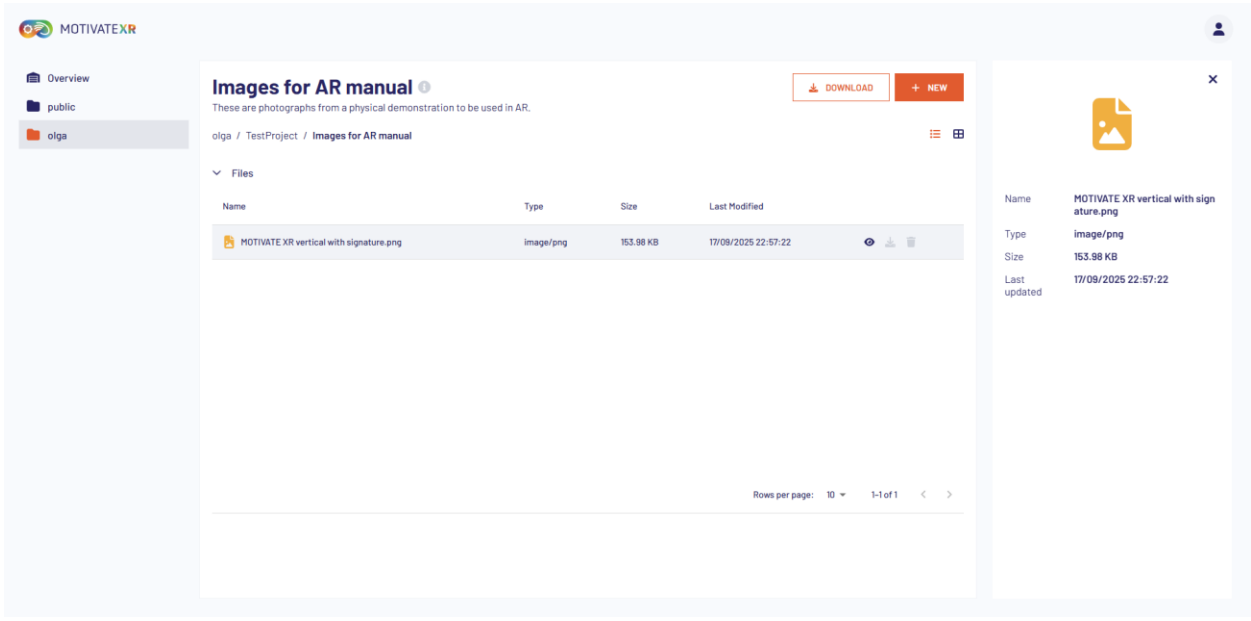

FIGURE 12 FILE METADATA VIEW

FIGURE 13 IMAGE FILE PREVIEW

FIGURE 14 3D MODEL PREVIEW

3.3. GROUP MANAGEMENT

Groups provide a way to organize users and control access to shared resources within the platform. In this beta release, group creation is restricted: only platform-wide administrators can create new groups and assign their initial administrator.

3.3.1. GROUP ADMINISTRATION

Once a group is created, the group admin manages its membership and roles:

- Invite Users: add new members to the group.
- Assign Roles: change a member's role between *viewer*, *editor*, and *admin*.
- Admin Restrictions: the original group admin cannot remove themselves from the admin role.

3.3.2. ROLES WITHIN A GROUP

- Admin: full control over the group, its members, and files.
- Editor: can create, update, and delete group files.
- Viewer: can browse and download files but cannot modify them.

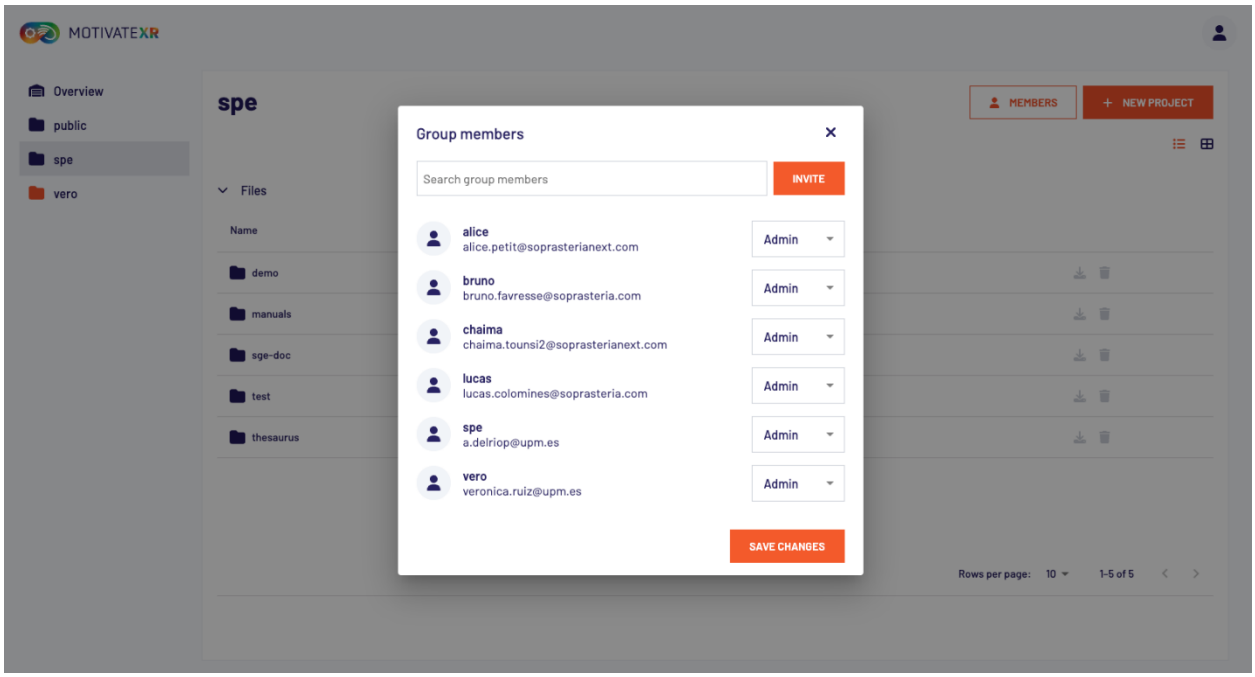

FIGURE 15 GROUP MEMBER MANGMENT

3.4. XR AUTHORING

Depending on the type of project that they have created, users can access integrated XR authoring tools.

Level of Immersion	XR authoring tools
VR	KAYROX, INSCAPE
AR	KAYROX
MR	NARRATIVE EDITOR

TABLE 1 XR AUTHORING TOOLS

As described in D6.1, each tool integrates with the platform in different ways. Specifically, in case of local authoring tools, like NARRATIVE EDITOR and INSCAPE, users will be forwarded with hyperlinks. Web-based authoring tools like KAYROX can have deeper integration with automated forwarding to the respective project environment across KAYROX and MOTIVATE XR.

3.5. XR EXPERIENCING

The MOTIVATE XR user interface is a web-based environment with features recommended for interaction using flat monitors and traditional input devices, i.e., mouse and keyboard, and not recommended for immersive environments. The MOTIVATE XR back-end service though can be used during experiencing from the respective XR compatible devices. To facilitate the user log-in process in the XR players, users can use the QR log-in feature to avoid manually typing with virtual keyboards or other methods and allow easy access to the MOTIVATE XR platform back-end services.

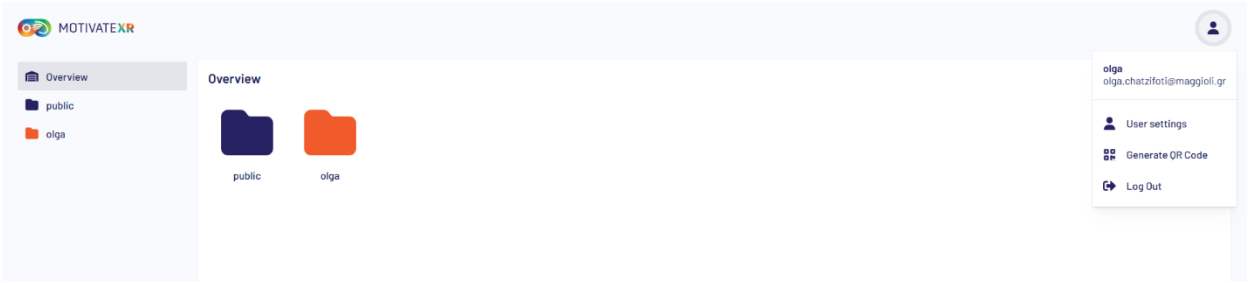

FIGURE 16 GENERATE QR OPTION

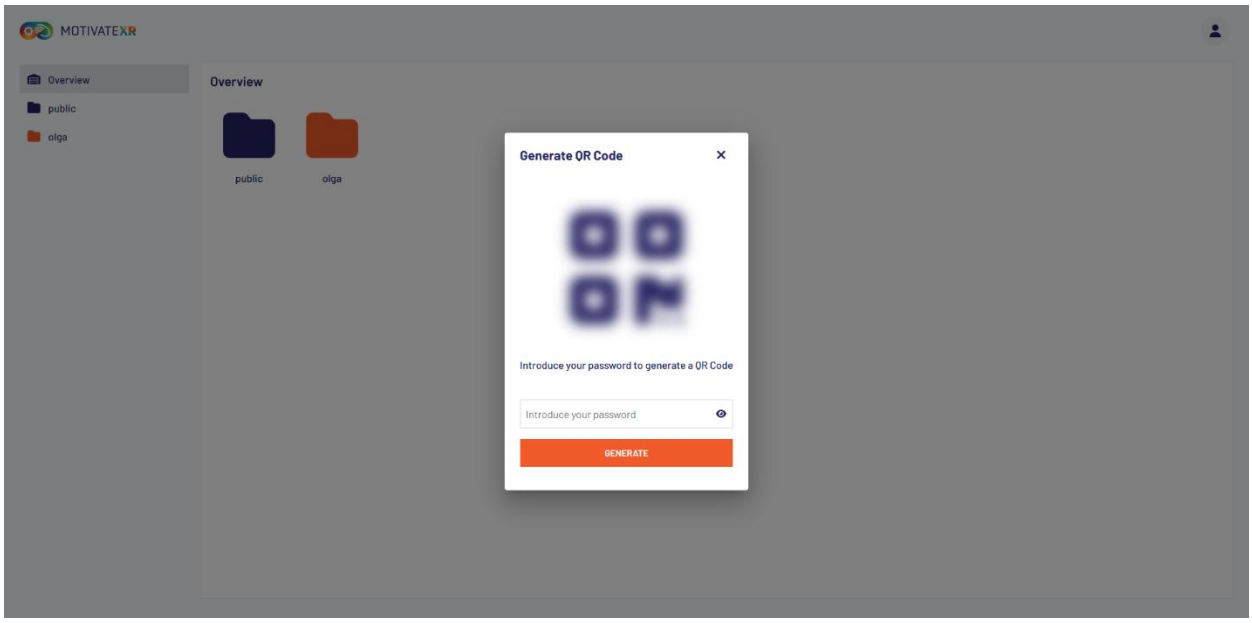

FIGURE 17 GENERATE QR PROCESS

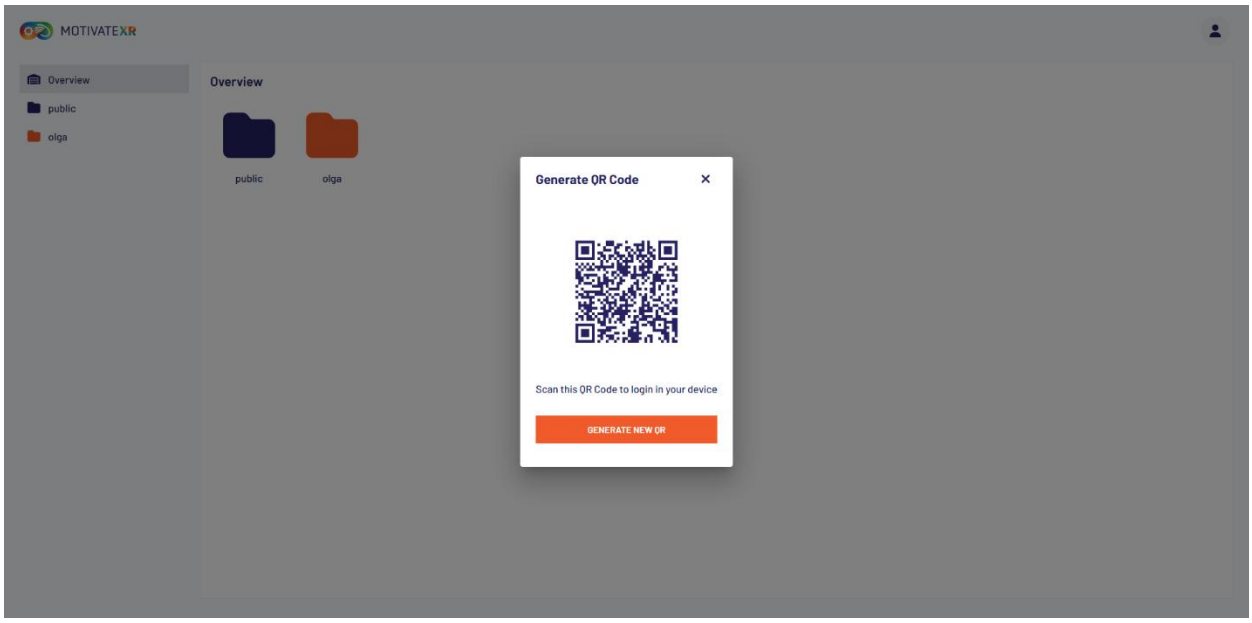

FIGURE 18 GENERATE QR RESULT

Advanced Processing Services

3.6. ADVANCED PROCESSING SERVICES

MOTIVATE XR services offer access to useful utilities for the authoring of XR experiences. These include the "video to 3d" service, which uses videogrammetric process to produce textured 3d

models from videos, and the semantic processing engine, which creates a conversation agent with a digest of provided pdfs.

3.6.1. VIDEO TO 3D SERVICE

In the “new” dropdown menu, a contextual button titled “Process 3D model from video” appears in folders with video content. The user can edit the name of the file to be created, the mesh quality or point cloud quality they desire for the output and one or more output formats. Output formats ply, obj, fbx and glb include mesh data, while output format las includes point cloud data. The user can select one or more videos to include in the videogrammetric processing.

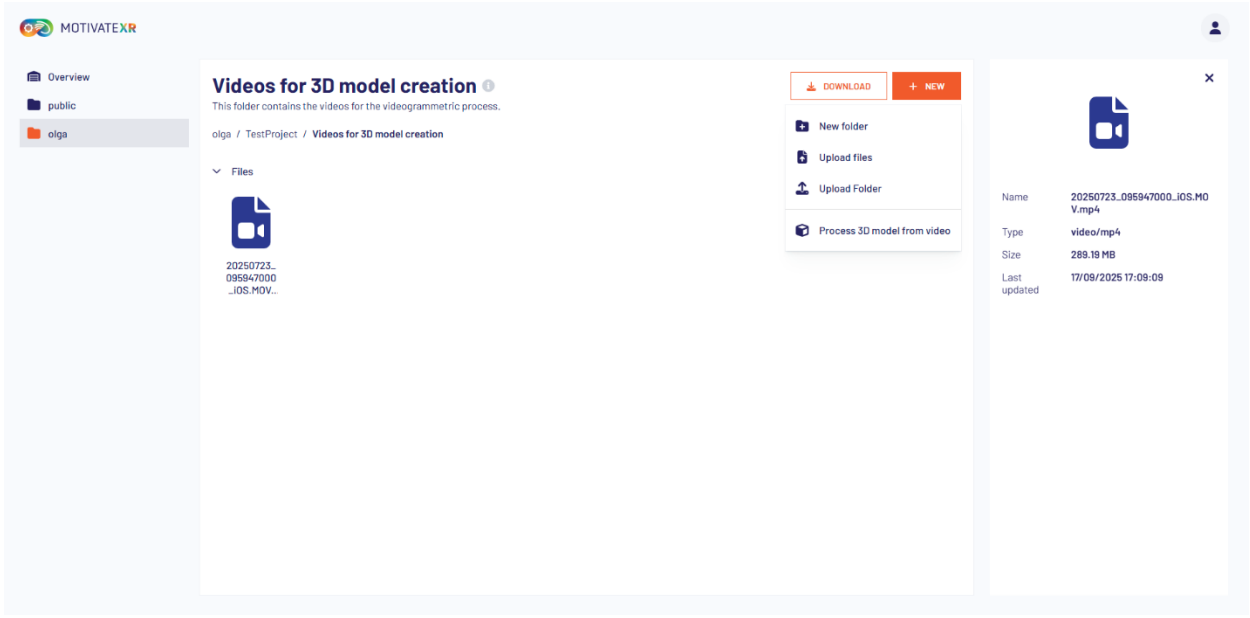

FIGURE 19 PROCESS 3D MODEL FROM VIDEO OPTION

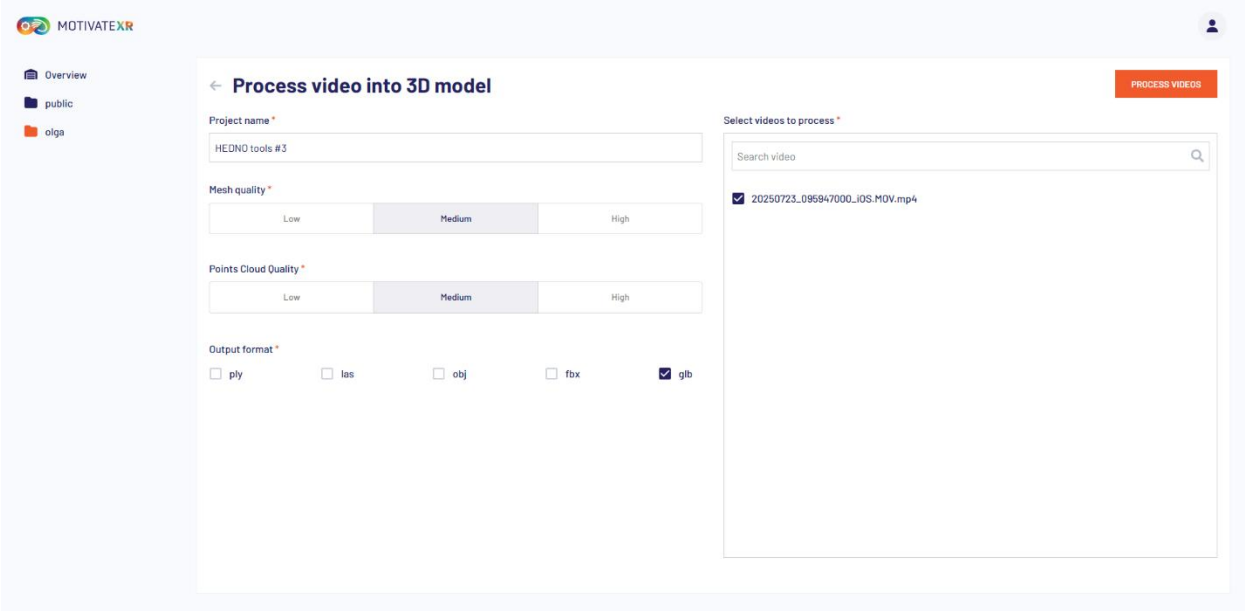

FIGURE 20 VIDEO PROCESSING PARAMETERS

After sending a request, a new panel appears in the interface with an updated display of the current progress of the videogrammetric task. Upon completion of the task, the user can download the output file.

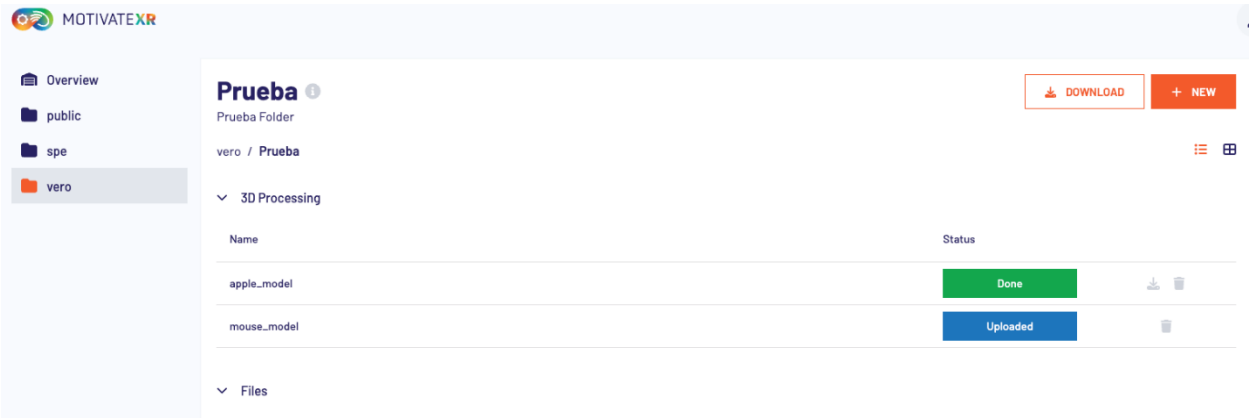

FIGURE 21 3D MODEL PROCESSING STATE

3.6.2. SEMANTIC PROCESSING ENGINE

When selecting pdf files, a contextual side panel appears with the metadata of the file and a user interface element to initiate semantic processing. The user can select one of the existing ones or create a new context. The context provides the conversation agent with background information to process the pdf contents. In the next version, a user will be able to create their own context by providing relevant information in a structure.

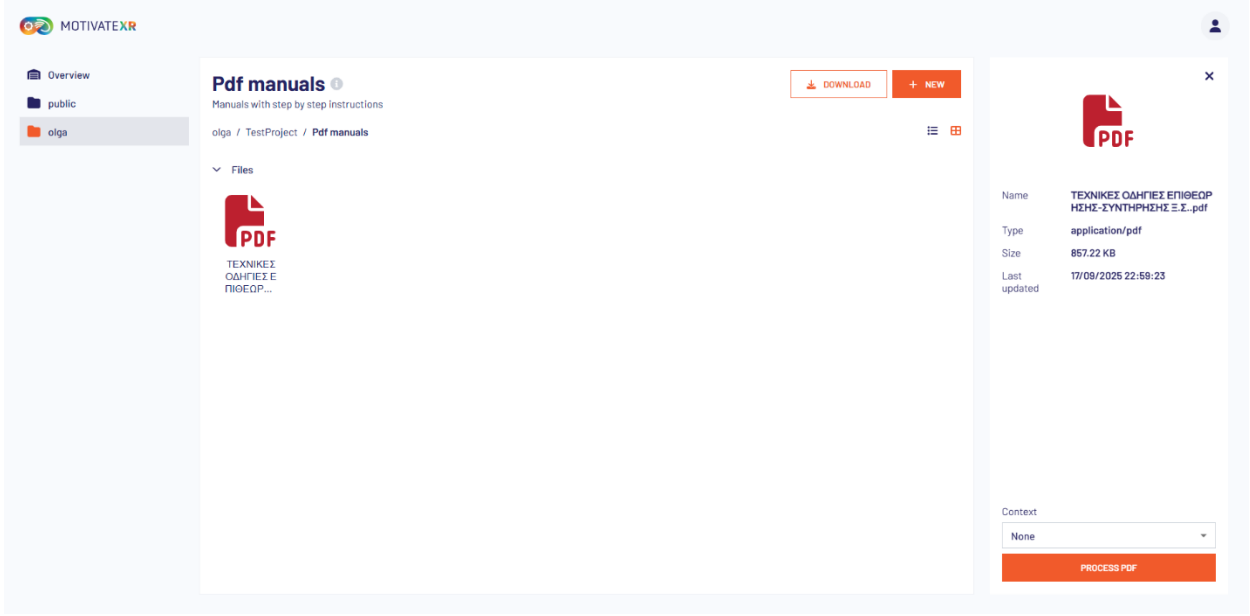

FIGURE 22 PDF PROCESS CONTEXTUAL MENU

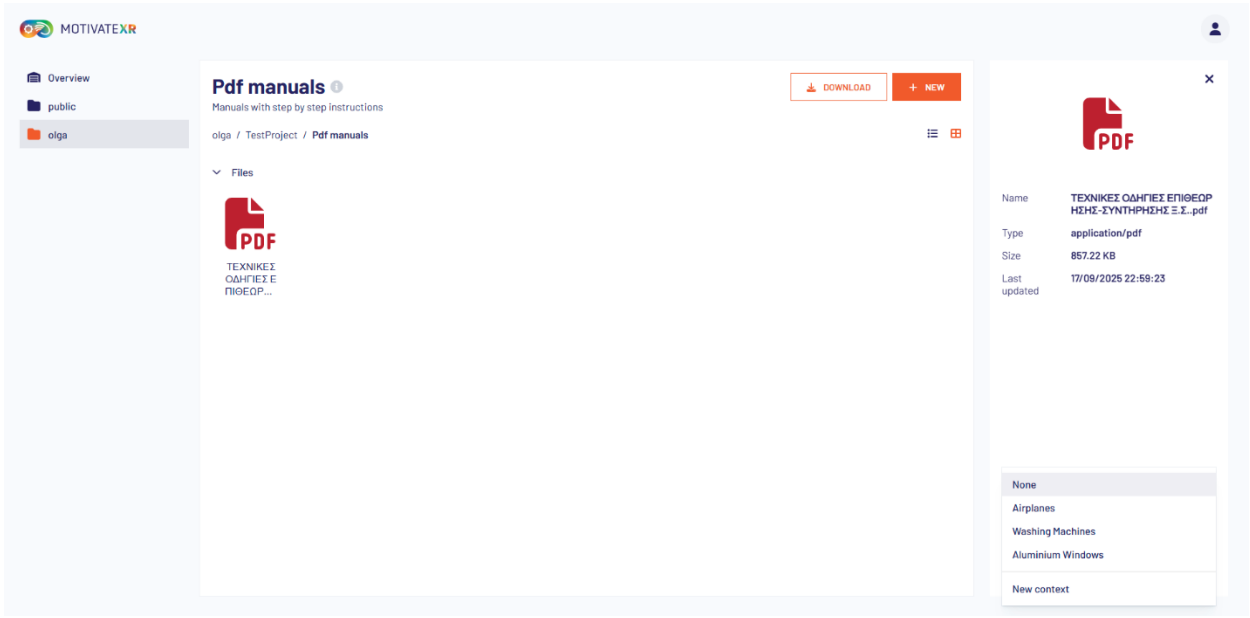

FIGURE 23 PROCESS CONTEXT

After processing is completed, the user can select the Q&A button to open a new interface, where they can interact with a conversation agent and preview the content of the pdf using natural language questions and answers.

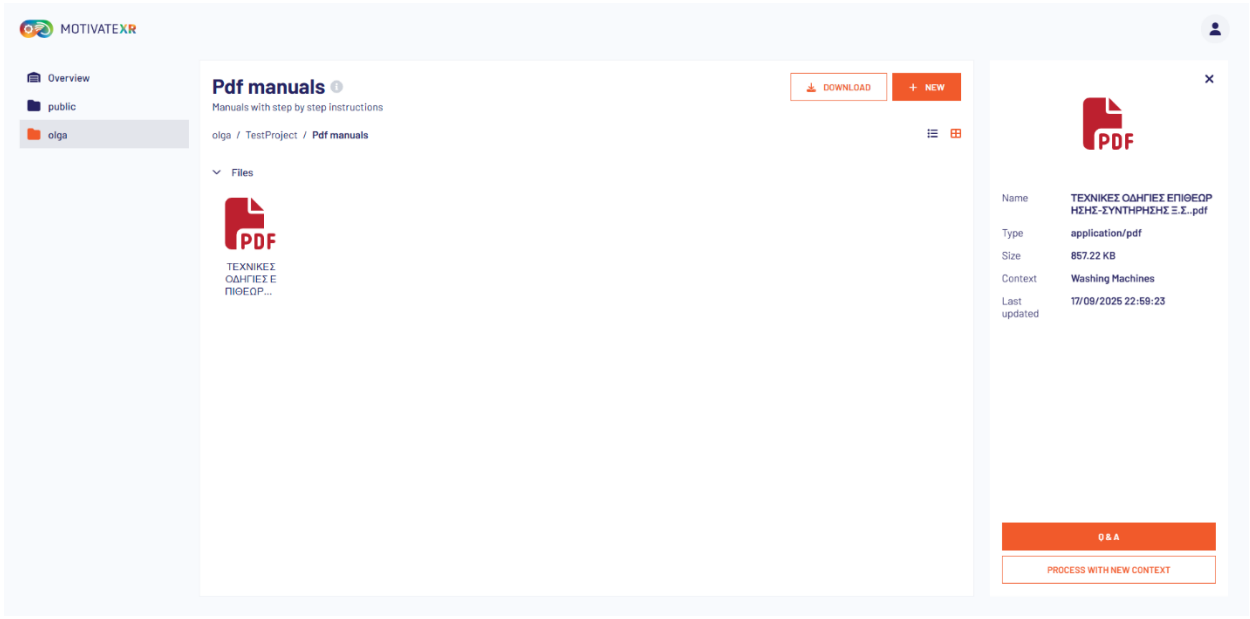

FIGURE 24 PDF Q&A OPTION

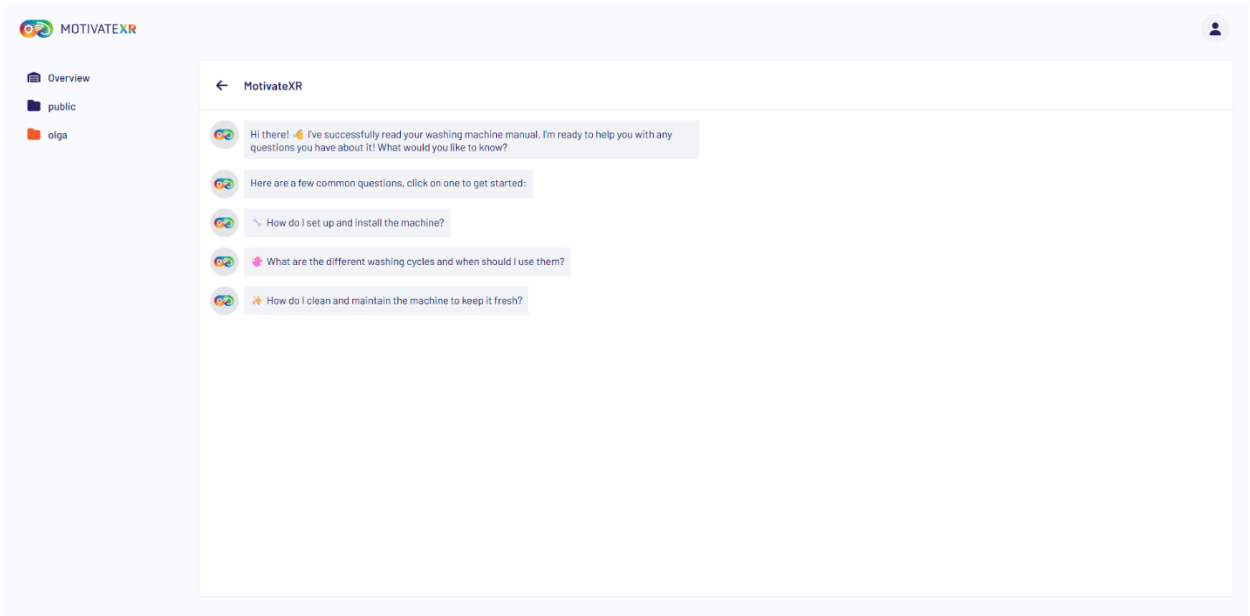

FIGURE 25 Q&A PREVIEW #1

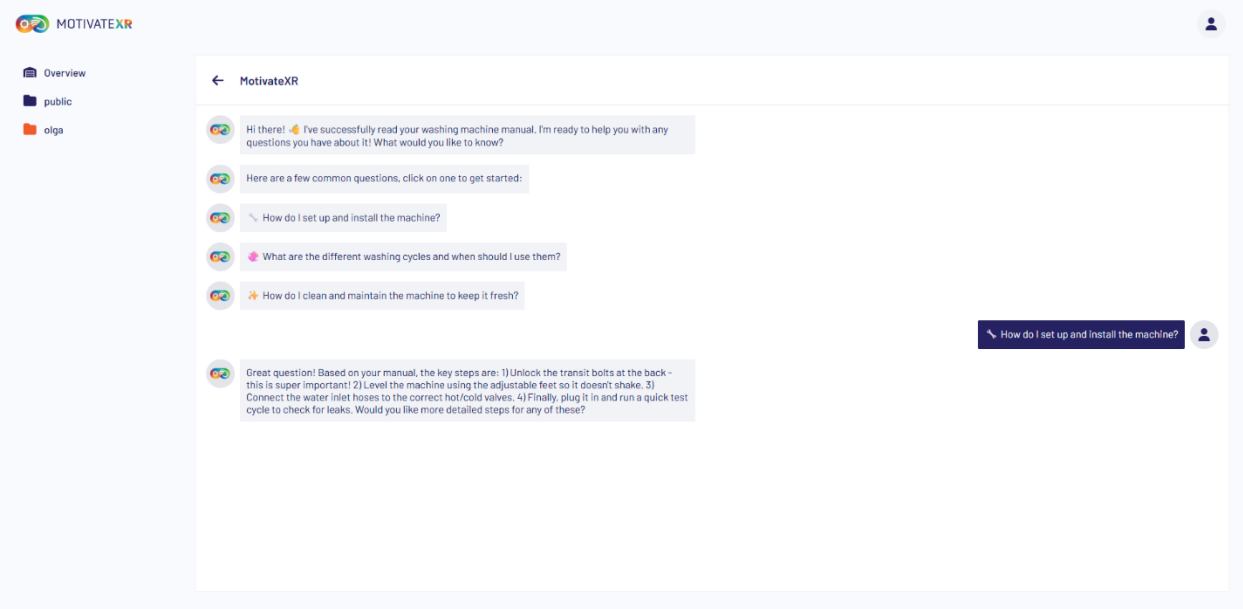

FIGURE 26 Q&A PREVIEW #2

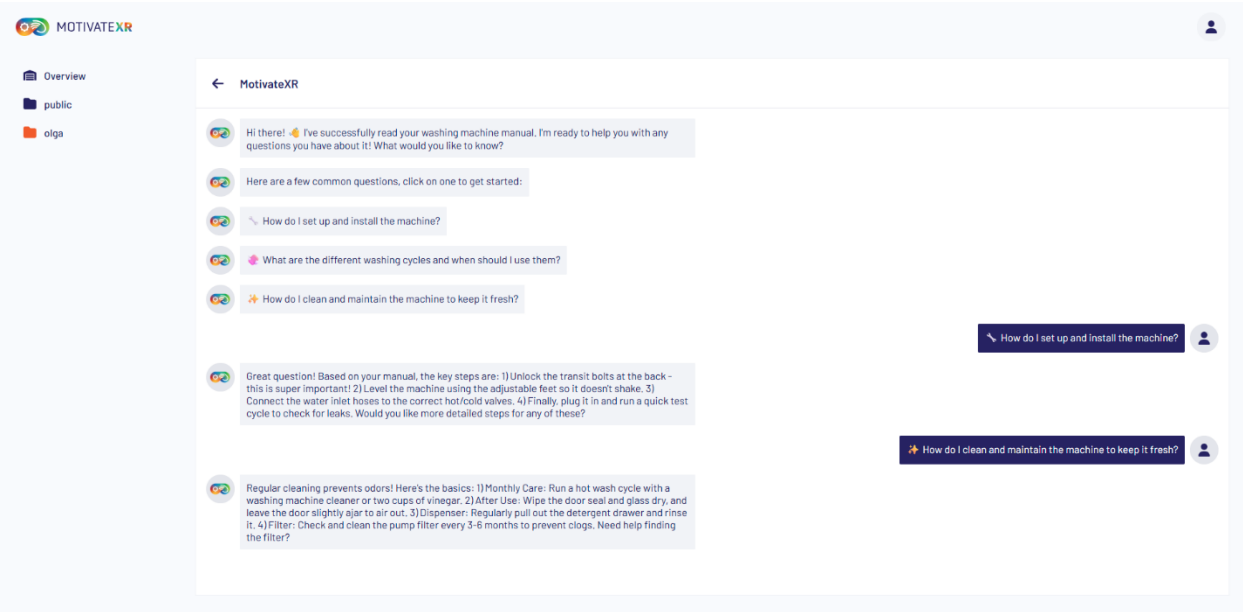

FIGURE 27 Q&A PREVIEW #3

4. ADMINISTRATIVE CAPABILITIES

4.1. ADMIN GUIDE: KEYCLOAK DASHBOARD

Keycloak includes a built-in Admin Console where administrators can manage authentication and authorization. In this platform, only users with the admin role can log in to the console.

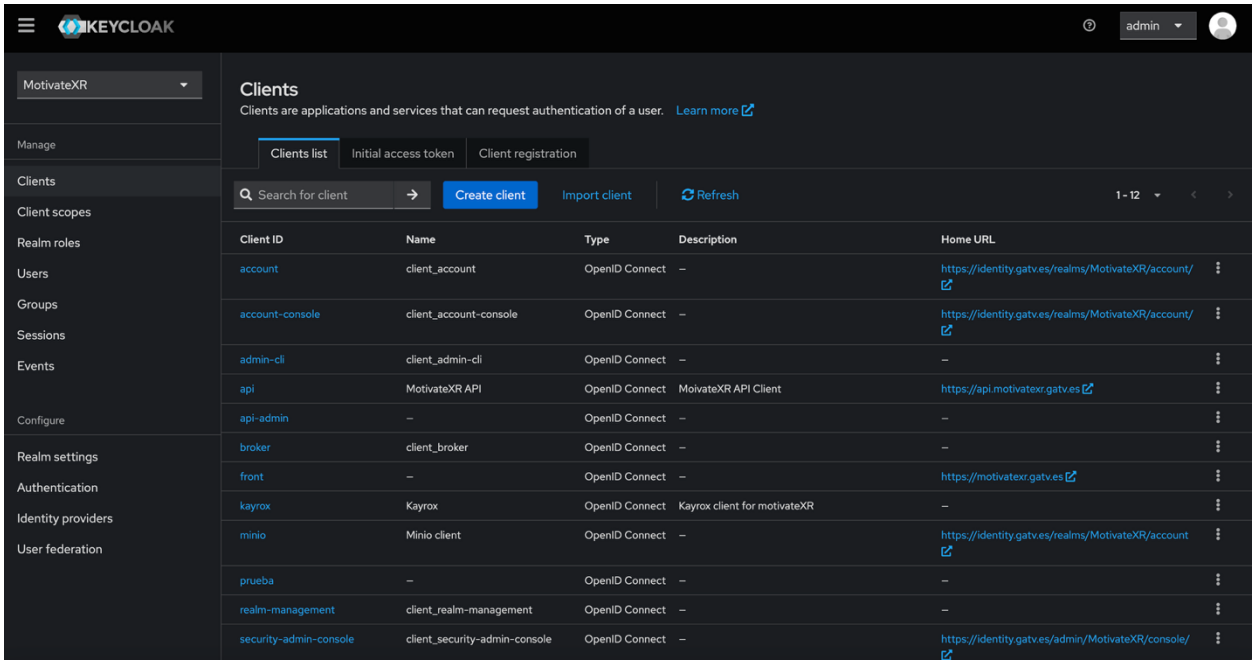

FIGURE 28 KEYCLOAK CLIENT MANAGEMENT

4.1.1. ACCESSING THE DASHBOARD

1. Open the Keycloak ingress URL in a browser.
2. Log in with your admin credentials.
3. You will be redirected to the Keycloak Admin Console.

4.1.2. COMMON ADMIN TASKS

1. Manage Users
   - Create new users and set their initial password.
   - Reset passwords and enforce password updates.

- Assign roles (admin, editor, viewer).

2. Manage Groups
   - Create new groups.
   - Add or remove members.
   - Set default roles for group members (default is viewer).
3. Manage Clients
   - View and configure clients (frontend, API, external tools).
   - Update redirect URIs or credentials for external integrations.
4. Token Settings
   - Adjust access token lifetimes (default: 30 minutes).
   - Enable/disable refresh tokens.

4.2. ADMIN GUIDE: MINIO DASHBOARD

MinIO provides a web dashboard for managing object storage. Admins can use it to create and monitor buckets, users, and access policies. Authentication to the dashboard uses the Keycloak integration, so only logged-in admins can make changes.

4.2.1. ACCESSING THE DASHBOARD

1. Open the MinIO ingress URL in a browser.
2. Log in with your Minio admin account.
3. You will be redirected to the MinIO Console.

4.2.2. COMMON ADMIN TASKS

1. Manage Buckets
   - View the three system buckets: public, private, and groups.
   - Create new group folders if needed.
   - Review bucket usage and storage statistics.
2. Manage Policies

- Policies are linked with Keycloak roles:
    - Admin → Full control.
    - Editor → Read/write.
    - Viewer → Read-only.
- Update or extend policies for specific buckets.

3. Monitor Usage
    - Track object counts and storage space.
    - Audit recent uploads/downloads.
4. Encryption and Security
    - Verify that default MinIO encryption is enabled for all data at rest.
    - Configure advanced options if required in future releases.

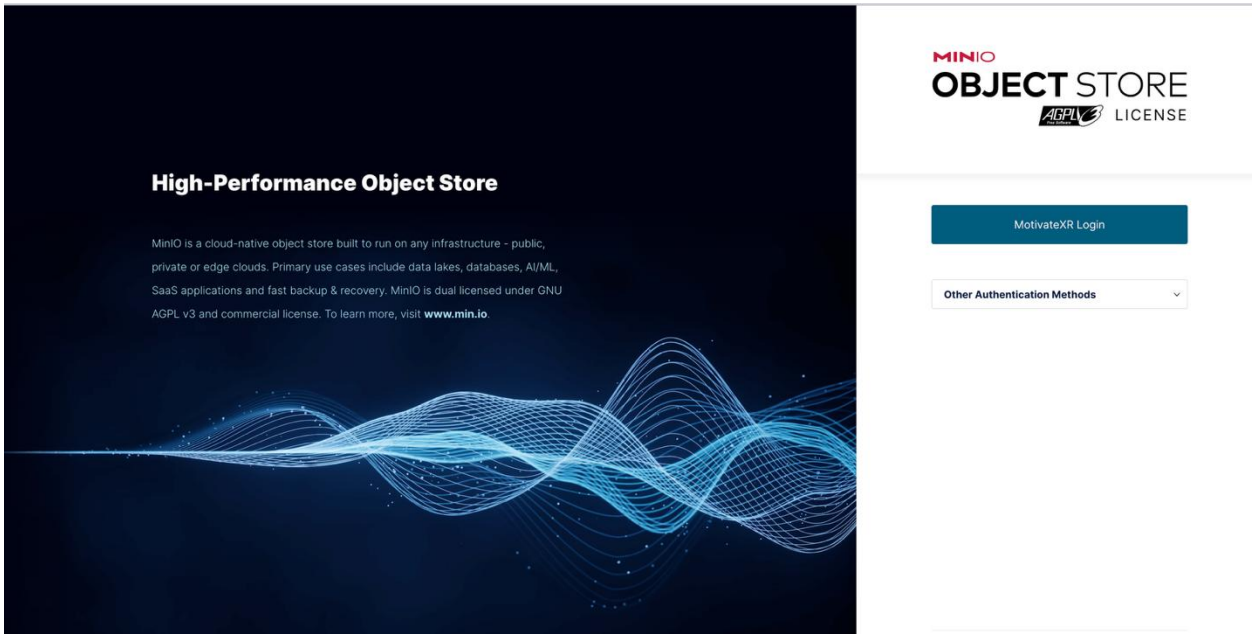

FIGURE 29 MINIO LOGIN

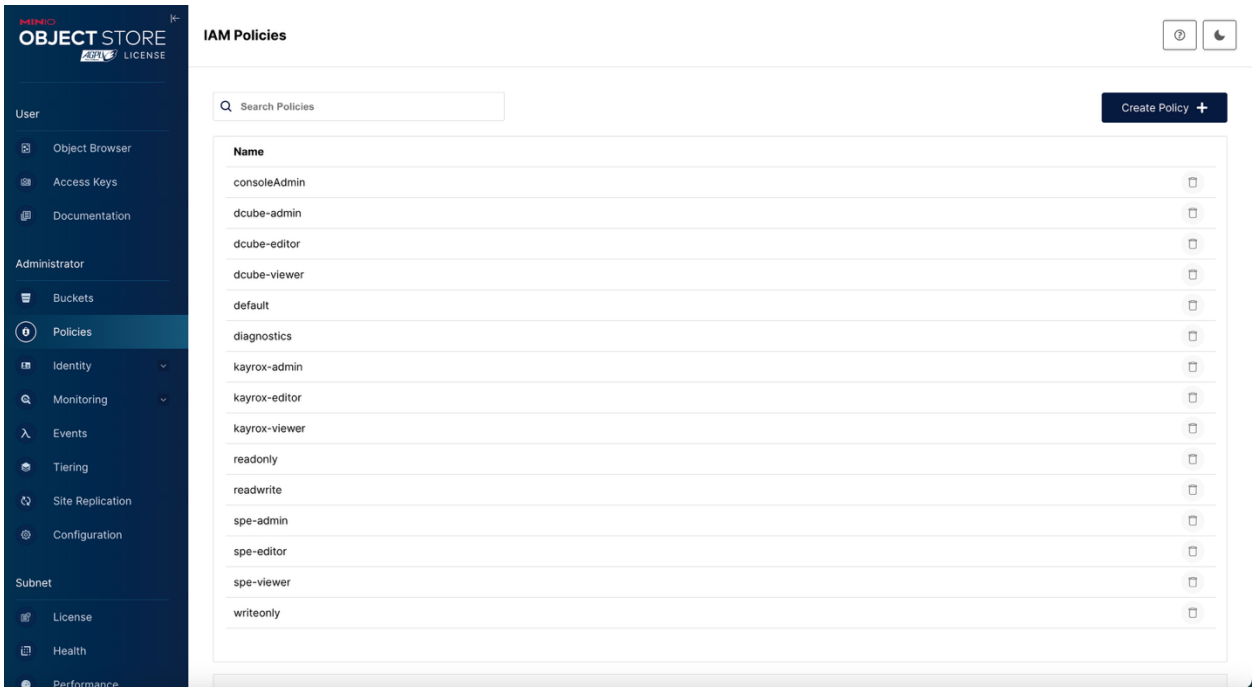

FIGURE 30 MINIO POLICY MANAGEMENT

5. FUTURE DEVELOPMENT

5.1. CURRENT STATE SUMMARY

This beta release (v1) establishes the foundational technical infrastructure for the MOTIVATE XR platform and successfully demonstrates integration of all work package deliverables from WP4, WP5, and WP6. The platform provides a complete XR development ecosystem that supports the full content lifecycle from creation through deployment and experiencing.

The current implementation achieves all core objectives for the beta phase:

- Complete architectural integration of all platform components
- Functional user management with role-based access control
- File storage and organization system with security policies
- Integration pathways for XR authoring tools (KAYROX, INSCAPE, NARRATIVE EDITOR)
- Advanced processing services (video-to-3D conversion and semantic processing engine)
- QR-code authentication for seamless XR device integration
- Administrative tools for platform management

5.2. ROADMAP FOR D6.4 MOTIVATE XR INTEGRATED RELEASE V2

The next release (D6.4 MOTIVATE XR Integrated Release v2, due M32) will focus on production readiness enhancements, including addressing user feedback, expanded service integrations, high-availability deployment configurations, and advanced monitoring capabilities. The platform will continue evolving to support emerging XR technologies and use cases across various industrial domains.

The development roadmap for v2 will be closely informed by feedback and requirements gathered during WP7 pilot sessions. This iterative approach ensures that the platform evolution directly addresses real-world usage scenarios and stakeholder needs. Key areas of focus for pilot feedback integration include:

- User experience improvements based on pilot participant feedback
- Performance optimizations identified during pilot testing
- Feature requirements emerging from real-world usage scenarios
- Security and compliance requirements from industrial pilot isolated environments

5.3. DEVELOPMENT TIMELINE

The development of MOTIVATE XR Integrated Release v2 follows a structured timeline aligned with project milestones:

- M16-18: User testing in the context of WP7
- M19-21: Feedback analysis and requirements extraction
- M25-28: Agile development iterations
- M29-30: Structured testing
- M31-32: Deployment, documentation, and final release preparations

This phased approach ensures continuous alignment between platform development and pilot activities, maximizing the value and applicability of the final release for industrial XR applications.

The beta release documented herein provides a robust foundation for this evolution, demonstrating successful technical integration while establishing the groundwork for production-scale deployment and expanded functionality in the comprehensive XR development ecosystem envisioned by the MOTIVATE XR project.