



MediaVerse

A universe of media assets
and co-creation opportunities

D6.4

Development, Integration and Testing of MediaVerse Components v2

Project Title	MediaVerse
Contract No.	957252
Instrument	Innovation Action
Thematic Priority	ICT-44-2020 Next Generation Media
Start of Project	1 October 2020
Duration	36 months

Deliverable title	Development, Integration and Testing of MediaVerse Components v2
Deliverable number	D6.4
Deliverable version	V1.0
Previous version(s)	D6.2
Contractual Date of delivery	30.06.2023
Actual Date of delivery	30.06.2022
Nature of deliverable	Demonstrator
Dissemination level	Public
Partner Responsible	ATC
Author(s)	Tasos Lampropoulos (ATC), Spyros Papafragkos (ATC)
Reviewer(s)	Stephan Gensch (VRAG), Sara De Luca (LINKS)
EC Project Officer	Luis Eduardo Martinez Lafuente

Abstract	D6.4 accompanies a report containing the updated system architecture based on the results of D2.3, as well as the final results from components integration and testing.
Keywords	MediaVerse software, platform, user guide, components, architecture, deployment

Copyright

© Copyright 2023 MediaVerse Consortium

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the MediaVerse Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.



MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252. The content of this document is © the author(s). For further information, visit mediaverse-project.eu.

Revision History

VERSION	DATE	MODIFIED BY	COMMENTS
V0.1	12/05/2023	Tasos Lampropoulos, Spyros Papafragkos (ATC)	First Draft Table of Contents
V0.2	16/05/2023	Tasos Lampropoulos (ATC)	Input in Section 2
V0.3	19/05/2023	Tasos Lampropoulos (ATC)	Input in Section 3
V0.4	22/05/2023	Spyros Papafragkos (ATC)	Input in Section 4
V0.5	26/05/2023	Tasos Lampropoulos (ATC)	Input in Section 5
V0.6	29/05/2023	Spyros Papafragkos (ATC)	Input in Section 1 and 6
V0.7	31/05/2023	Tasos Lampropoulos, Spyros Papafragkos (ATC)	Input in all Sections
V0.8	23/06/2023	Stephan Gensch (VRAG), Sara De Luca (LINKS)	Internal review
V0.9	26/06/2023	Tasos Lampropoulos, Spyros Papafragkos (ATC)	Addressing comments
V1.0	30/06/2023	Evangelia Kartsounidou, Akis Papadopoulos	Final review and QA

Glossary

ABBREVIATION	MEANING
DAM	Digital Asset Management
DB	Database
GDPR	General Data Protection Regulation
IPFS	InterPlanetary File System
IPR	Intellectual Property Rights
MV	MediaVerse
SC	Smart Contract
SLC	Smart Legal Contract
UI	User Interface
WP	Work Package

Table of Contents

Revision History	3
Glossary	3
Index of Figures	6
Index of Tables.....	6
Executive Summary	7
1 Introduction.....	8
2 System Overview	9
2.1 Integration Status	9
2.1.1 KONG-GATEWAY – Gateway Service	10
2.1.2 MV-DAM-API – Digital Asset Management	10
2.1.3 APACHE SOLR – Index of Content Metadata	11
2.1.4 MONGO – Data Layer	11
2.1.5 IPR-SERVICE – Intellectual Property Rights Service	11
2.1.6 MV-SLC-ENGINE – Smart Legal Contracts Management	11
2.1.7 CICERO-SERVER – Smart Legal Contracts and Templating System.....	11
2.1.8 MV-BCSP - MV-BCSPEH - MV-ETH – Smart Contracts and Blockchain Management	11
2.1.9 SLC-TEMPLATE-STUDIO – Smart Contracts Template Studio	11
2.1.10 DASHBOARD-UI.....	12
2.1.11 IPFS-HOST and IPFS-API – Decentralized Framework.....	12
2.1.12 TRANSCODER and PUBLISHER – Content Adaptation Services	12
2.1.13 COPYRIGHT-NEGOTIATION – License Advisor Module.....	12
2.1.14 MODERATION-UI	12
2.1.15 CMRR – Retrieval and Recommendation System.....	12
2.1.16 FADER360-APPLICATION	12
2.1.17 MV_OMAF-APP.....	12
2.1.18 PROMETHEUS / GRAFANA – Monitoring Tools	13
2.2 Implemented Features	13
2.3 Deployment Status	15
3 Installation Instructions.....	17
3.1 Basic Configuration.....	17
3.1.1 Mongo.....	17
3.1.2 DAM.....	18
3.1.3 Rights-Management Components.....	22

3.1.4	UI.....	25
3.1.5	IPFS API	26
3.1.6	IPFS HOST.....	27
3.1.7	POSTFIX (Optional)	27
3.1.8	KONG GATEWAY	28
3.1.9	MODERATION UI.....	29
3.1.10	FADER	29
3.1.11	OMAF	29
3.1.12	Prometheus and Grafana (Optional)	30
3.2	Docker Execution.....	31
4	User Manual	32
5	Testing	46
5.1	Testing Plan	46
5.2	Test Cases	46
6	Conclusion	51

Index of Figures

Figure 1: Docker Containers	9
Figure 2: List with the available functionalities (part I)	13
Figure 3: List with the available functionalities (part II)	14
Figure 4: List with the available functionalities (part III)	15
Figure 5: Login webpage.....	32
Figure 6: Create account page.....	32
Figure 7: Dashboard page.....	33
Figure 8: Upload page.....	33
Figure 9: Fill details when uploading an asset.....	34
Figure 10: My Assets page	35
Figure 11: Assets details	36
Figure 12: License registration for an asset.....	37
Figure 13: Recommendations page	37
Figure 14: History page.....	38
Figure 15: Search page	39
Figure 16: Projects page	40
Figure 17: Project details page	41
Figure 18: VRodos login webpage	42
Figure 19: Fader login webpage	43
Figure 20: Secondary menu	43
Figure 21: Profile details.....	44
Figure 22: Moderator page.....	45

Index of Tables

Table 1: Servers' characteristics and the corresponding URLs.....	15
Table 2: Overview of the testing plan	46

Executive Summary

This document is the final report for the implemented, tested, and integrated MediaVerse (MV) platform demonstrator in its final release. It provides all the necessary details (e.g., Docker containers) concerning the status of the MV node v2.0. Since this deliverable is the updated version of D6.2 - Development, Integration and Testing of MediaVerse Components v1¹ some of the information included may have been repeated for the sake of completeness and clarity of the document. At the same time, we highlight the updated features and the new services. Specifically, this document describes the integrated components, the main functionalities, and an overview of the deployed nodes at the time of writing this report. It also includes installation instructions for helping any interested party to deploy their own MediaVerse node. Moreover, it provides a user guide along with screenshots that demonstrate the navigation in the platform and a set of key functionalities of this release. Finally, it presents an overview of the testing strategy implemented to verify the functionalities of the project. It outlines the test cases that have been created and provides detailed specifications of the testing plan.

¹ <https://mediaverse-project.eu/wp-content/uploads/2022/07/D6.2-v1.0.pdf>

1 Introduction

MediaVerse is a pioneering project that aims to revolutionize media asset management by providing a decentralized platform. The project partners have diligently carried out extensive research and development activities to deliver a fully functional platform by the final release. This update highlights the progress made in terms of platform conceptualization, user requirements, and implementation efforts.

The conceptual architecture of the MediaVerse platform was presented in deliverable D2.2 - Conceptual Design of the MediaVerse Framework². This deliverable outlined the foundational design principles and structural elements of the platform. It served as a guiding framework for subsequent development activities, ensuring that the final release aligns with the original vision.

Deliverable D2.1 - Use Cases and User Requirements³ played a crucial role in shaping the functionalities of the MediaVerse platform. Through meticulous analysis of user requirements, the project team identified key features and capabilities that would enhance media asset management. These requirements acted as the foundation for the platform's development, ensuring that it addresses the specific needs and pain points of media professionals and content creators.

The successful implementation of the MediaVerse platform has been the result of dedicated research conducted within the technical work packages (WP3, WP4, WP5, and WP6). These work packages focused on the actual implementation of various components, such as decentralized storage, metadata management, content discovery, and user authentication. The research conducted in these work packages informed the development process, ensuring the platform's technical feasibility and robustness.

As the project reaches its final release, the MediaVerse platform is now fully functional, embodying the collective efforts and expertise of the project partners. The platform offers an innovative, decentralized solution for media asset management, empowering users to securely store, organize, and discover media assets. It represents a significant leap forward in improving efficiency and collaboration within the media industry.

The following sections provide details about the status of the last MV platform release. More specifically, Section 2 presents a comprehensive explanation of all the integrated components in this final release, the available functionalities of these components, and information on the deployed nodes so far. Section 3 is a user manual that can be used as guide of the platform for prospective users. In Section 4, we describe the testing strategy, while we conclude in Section 5.

²https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D2.2_Conceptual-Design-of-the-MediaVerse-Framework.pdf

³<https://mediaverse-project.eu/wp-content/uploads/2021/04/D2.1-V1.0.pdf>

2 System Overview

2.1 Integration Status

A MediaVerse node consists of a set of Docker containers, as depicted in Figure 1. These contain core functionalities of each node developed in WP6, such as user authentication and media asset management, as well as services that have been developed in WP3, WP4 and WP5 and integrated in the MV node architecture (through WP6), such as federated search and retrieval, IPR management, and media asset publishing.

The services developed in WP3 for media annotation are not part of the platform but have been integrated as external services called by the MediaVerse Digital Asset management component (*mv-dam-api* in Figure 1) through an API gateway. In that way, the annotations for each uploaded asset are stored and indexed to facilitate the retrieval of assets. The work conducted in WP4 resulted in the different components under the IPR SERVICES group in Figure 1. The authoring tools part of WP5 are partly integrated into the platform, the Fader 360 platform can be optionally included in the MediaVerse node while it can also be deployed as a standalone application with or without MediaVerse functionality. The VRodos product is not part of the node but it remains an external service. More precisely, the current release provides all the needed mechanisms for user authentication, project management and asset management, retrieval, and publishing. In addition, all three core media types are supported (images, videos, 3D objects, and scenes), therefore authoring tools can communicate with MediaVerse nodes (through the *kong⁴-gateway*) and retrieve content to be used in authoring procedures. Finally, optionally monitoring tools have been included in the node so that the performance and the status of the services can be live monitored.

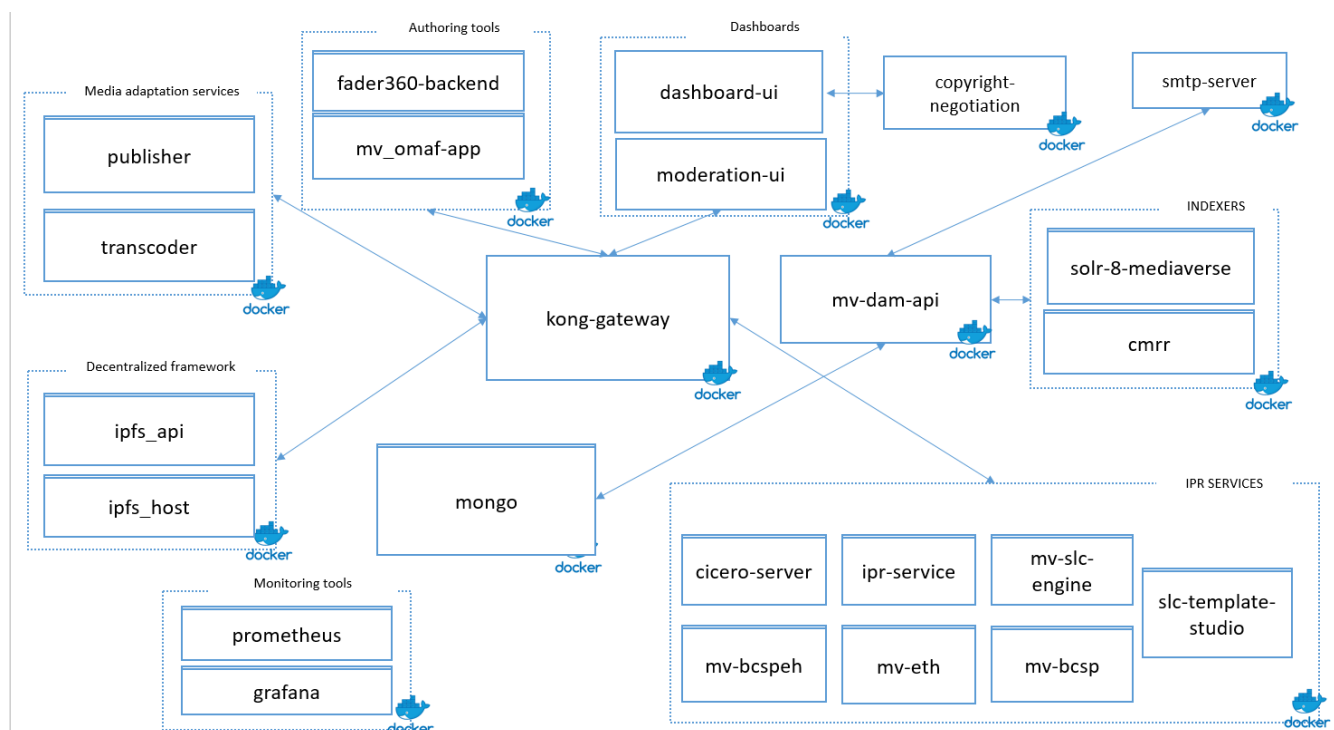


Figure 1: Docker Containers

⁴ <https://konghq.com/kong>

2.1.1 KONG-GATEWAY – Gateway Service

The gateway, based on kong, manages the incoming/outgoing traffic between the end-user and the MV platform, but also the communication between the rest of the services in the platform.

2.1.2 MV-DAM-API – Digital Asset Management

The Digital Asset Management is the central element of the MediaVerse node, which participates in almost all interactions between the platform components. It is responsible for the implementation of the business logic.

- User/authentication management
 - Registration
 - Authentication
 - Retrieve, Update user details
- Asset management
 - Upload
 - Update
 - Delete
 - Download
 - Search
 - Share images on Social Media
- Subtitle management
 - Create
 - Delete
 - Retrieve/Download
- Project management
 - Create
 - Update
 - Delete
 - Search
 - Manage owners and contributors
 - Retrieve Assets added to Projects
- Notification management
 - Creation
 - Update
 - Delete
 - Search
- IPR services management (SLC and SLC Template)
 - Creation
 - Update
 - Delete
 - Search
 - Proxy between UI and IPR services
- Recommendation management
 - Retrieve contents
 - Retrieve recommendations

- Moderation management
 - Manage rules
 - Retrieve moderated Assets
 - Publish/un-publish Assets

2.1.3 APACHE SOLR – Index of Content Metadata

The module is responsible for indexing asset details from different sources and making them searchable in near real-time. During the ingestion of an asset, the DAM will send all the available metadata to the Apache Solr instance for the latter to index it and make it available for search.

Apart from the metadata provided by the user during upload such as description, and metadata such as upload time and media type, AI-based annotations are also added to support visual content-based search. In particular, the AI-based annotations that are automatically detected in the image or video content of the assets and can be searched include the following: a) Celebrities (images, videos), b) Actions (images, videos), c) automatically generated caption (images), d) depicted objects (images, videos, 3D scenes), e) meme class (images), f) disturbing and NSFW flags (image, videos).

2.1.4 MONGO – Data Layer

This component is used by the DAM to persist users, assets, projects, and all related data to support MV usage.

2.1.5 IPR-SERVICE – Intellectual Property Rights Service

This API handles the interactions with the Blockchain Service Provider and the Smart Legal Contract (SLC) Engine.

2.1.6 MV-SLC-ENGINE – Smart Legal Contracts Management

This engine handles all the run-time features related to SLCs (e.g., the creation of SLC instances, check for the trigger of specific SLC-related events).

2.1.7 CICERO-SERVER – Smart Legal Contracts and Templating System

This template engine handles the Smart Legal Contract Templates that conform to the [Accord Project Template Specification](#).

2.1.8 MV-BCSP - MV-BCSPEH - MV-ETH – Smart Contracts and Blockchain Management

These services handle the deployment and the management of the Smart Contracts (SCs) and the interaction with the blockchain in general, ensuring the notarization of SLCs, and the implementation of the blockchain SCs counterparts of the rights on digital assets in a way that is transparent to the user.

2.1.9 SLC-TEMPLATE-STUDIO – Smart Contracts Template Studio

Based on the Accord Project technology, this template studio fork lets the users to load, edit and test clause or contract templates.

2.1.10 DASHBOARD-UI

The web-based user interface enables users to access the different services of MediaVerse in an intuitive way. The dashboard functionality focuses on user, content, and project management, offering support to image, audio, video, and 3D content.

2.1.11 IPFS-HOST and IPFS-API – Decentralized Framework

These services allow users to search for content in the entire MediaVerse network without the need to know where and how many nodes are in the network. This service forwards the search queries and aggregates the search results across the network.

2.1.12 TRANSCODER and PUBLISHER – Content Adaptation Services

This service automatically generates preview formats from media files, such as GIF files, thumbnails, audio watermarks, etc. In accordance with user access rights, these previews are used during search and retrieval to facilitate content browsing and discovery from users that do not have access to the actual content but need a preview of the asset they intend to purchase. It also produces adaptive streaming content from video files.

2.1.13 COPYRIGHT-NEGOTIATION – License Advisor Module

The license advisor module is embedded in the UI to recommend appropriate licenses for asset media files.

2.1.14 MODERATION-UI

This interface allows node admins to configure the moderation rules, which will be used during the retrieval of Assets. Furthermore, admins can retrieve all the moderated Assets and publish or unpublish them.

2.1.15 CMRR – Retrieval and Recommendation System

This service indexes all the published Assets and returns recommended content per user according to the Assets that are uploaded by each user.

2.1.16 FADER360-APPLICATION

Fader is a web-based platform for storytellers who want to share their 360° stories with the world. It provides users with the ability to combine 360° content with interactive hotspots to create immersive virtual reality experiences.

2.1.17 MV_OMAF-APP

This service is used to produce OMAF⁵ compliant adaptive streaming 360-degree video.

⁵ <https://mpeg.chiariglione.org/standards/mpeg-i/omnidirectional-media-format>

2.1.18 PROMETHEUS / GRAFANA – Monitoring Tools

Prometheus⁶ is an open-source systems monitoring and alerting toolkit. The metrics collected by the Prometheus tool are visualized in the Grafana⁷ dashboards. Those services are not exposed to the Internet and their use is intended only for the administrators of the node.

2.2 Implemented Features

This section presents the available features/functionalities of the MV node at the time of writing. Figures 2, 3 and 4 are screenshots from the tool used by the involved partners for issue monitoring. Further details on specific features can be found in the deliverable D2.3 - Use Cases and User Requirements.

PUBL-02 Share content in SoMe (ATC)	#109 · created 1 year ago by Administrator	Available for v2.0	Pilots	Publishing	UC 1	UC 2	UC 3
MDRT-05 Comedy/satire flag (ATOS, ATC)	#104 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots			
ANNO-01 Automatic tagging (CERTH)	#103 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots	UC 1	UC 3	
SRCH-01 Search across nodes (ATOS)	#100 · created 1 year ago by Administrator	Available for v2.0	Consumption	Creation	Pilots	UC 1	UC 3
SRCH-03 Search external sources through MV search (ATC)	#98 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots	UC 1	UC 3	
SRCH-04 Save search (ATC)	#97 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots			
SRCH-05 Turn search to Call for Content (ATC)	#96 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots			
SRCH-06 Search filters (LINKS)	#95 · created 1 year ago by Administrator	Available for v2.0	Consumption	Creation	Pilots	UC 1	UC 3
RGHT-01 Select licences (TLX)	#93 · created 1 year ago by Administrator	Available for v2.0	Pilots	Publishing	UC 1	UC 2	UC 3
RGHT-02 Guidelines for licence selection (TLX, ATOS)	#92 · created 1 year ago by Administrator	Available for v2.0	Pilots	Publishing	UC 1	UC 2	UC 3
RGHT-03 Trace content sources (ATC)	#91 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots	UC 1	UC 3	
RGHT-05 Moral rights per contributed segment (FIN/TLX)	#89 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots	UC 1	UC 2	

Figure 2: List with the available functionalities (part I)

⁶ <https://prometheus.io/docs/introduction/overview/>

⁷ <https://grafana.com/grafana/>

RGHT-06 Invite external creators (ATOS)	
#88 · created 1 year ago by Administrator	Administration Available for v2.0 Pilots UC 2 UC 3
TRSL-01 Assets' short descriptions (ATOS)	
#87 · created 1 year ago by Administrator	Administration Available for v2.0 Consumption Pilots UC 1 UC 3
TRSL-02 Subtitles in many languages (STXT)	
#86 · created 1 year ago by Administrator	Available for v2.0 Consumption Creation Pilots UC 1 UC 2 UC 3
XR-08 User presence in VR (Webcam) (CERTH)	
#84 · created 1 year ago by Administrator	Available for v2.0 Consumption Pilots UC 3
XR-09 interfaces for VR devices (CERTH)	
#83 · created 1 year ago by Administrator	Available for v2.0 Consumption Pilots UC 3
XR-03 Import 3D objects (CERTH)	
#81 · created 1 year ago by Administrator	Available for v2.0 Creation Pilots UC 3
XR-04 Decorate VR walls (CERTH)	
#80 · created 1 year ago by Administrator	Available for v2.0 Creation Pilots UC 3
XR-06 Hotspots in 360 video (VRAG)	
#78 · created 1 year ago by Administrator	Available for v2.0 Creation Pilots UC 1 UC 2

Figure 3: List with the available functionalities (part II)

AUTH-04 Multilingual authoring (VRAG)	#75 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots	UC 1	UC 2	UC 3	
AUTH-05 Subtitles (STXT)	#74 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots	UC 1	UC 2	UC 3	
BRKR-02 Call for content (ATC)	#68 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots				
PLAY-04 Consumption in HMD, HbbTV (ATOS)	#66 · created 1 year ago by Administrator	Available for v2.0	Consumption	Pilots				
PLAY-04 Consumption on PC, mobile (ATOS)	#65 · created 1 year ago by Administrator	Available for v2.0	Consumption	Pilots	UC 1	UC 2	UC 3	
COLL-02 Project overview (ATOS)	#58 · created 1 year ago by Administrator	Available for v2.0	Creation	Pilots	UC 1	UC 2	UC 3	
UX-02 Content information (ATOS)	#54 · created 1 year ago by Administrator	Administration	Available for v2.0	Consumption	Pilots	UC 1	UC 2	UC 3
USMG-04 Profile dashboard (ATOS)	#53 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots	UC 1	UC 2	UC 3	
USMG-03 Edit profile information (ATOS, ATC)	#52 · created 1 year ago by Administrator	Administration	Available for v2.0	Pilots	UC 1	UC 2		
USMG-01: Register/Log in (ATC)	#49 · created 1 year ago by Spyros Papafragkos	Administration	Available for v2.0	Pilots	UC 1	UC 2	UC 3	
Upload Content (360 video) (ATOS)	#48 · created 1 year ago by Spyros Papafragkos	Available for v2.0	Creation	Pilots	UC 1	UC 2	UC 3	
RGHT-02 Adding copyright information (ATOS)	#46 · created 1 year ago by Spyros Papafragkos	Available for v2.0	Pilots	Publishing	UC 1	UC 2	UC 3	
AUTH-META-02 Fader SaaS (VRAG)	#38 · created 1 year ago by Administrator	MS3: First Platform Release	Asset Retrieval	Available for v2.0	Creation	Publishing	UC 1	UC 2

Figure 4. List with the available functionalities (part III).

2.3 Deployment Status

The MediaVerse software management is realized using a semantic versioning technique for each component that is integrated in the MV node. Every version of each module is stored in a dedicated docker hub [repository](#) as a docker image. The latest versions of the above images are then included in the docker compose configuration file that contains the instructions for creating all the available resources for the MediaVerse node.

Whenever a new version of a component is available, it is dockerized, uploaded in the docker hub and a new merge request in the respective Gitlab [repo](#) is triggered for the update of the docker compose file. In that way, the docker compose configuration is reviewed and it is available for cloning in the MediaVerse nodes.

The latest platform version has been deployed in three different nodes and specifically on the premises of ATC, CERTH and ATOS. Table 1 provides the deployment details.

Table 1: Servers' characteristics and the corresponding URLs

NODE	SPECIFICATIONS			URL
	OS	CPU	RAM	
ATC	20.04.1-Ubuntu	Intel(R) Xeon(R) CPU E5-2620 v2	8 GB	https://dashboard.mediaverse.atc.gr/

		@ 2.10GHz (2 cores)		
CERTH	20.04.4-Ubuntu LTS	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz (24 cores)	128 GB	https://mediaverse.mever.gr/
ATOS	20.04.3-Ubuntu LTS	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz (8 cores)	16 GB	https://mediaverselab.ari-imet.eu/

3 Installation Instructions

Here, we detail all the necessary steps needed for the installation of the MV node.

3.1 Basic Configuration

The deployment structure consists of a **docker-compose** file, a **mongo init script** and a **config folder**. The docker-compose file, the init-mongo.js and the config folder must be placed in the root level.

1. The docker-compose file contains all the docker images, networks and volumes needed for the docker engine to deploy the MV node. It is predefined and it is common for all nodes, but node administrators can edit it accordingly to change the port in which each service is exposed.
2. The mongo init script will be called by the docker-compose file to properly initialize the DB. It is predefined and it is common for all the nodes.
3. The config folder has the following structure:

```
config
  ..dam
  ..fader
  ..ipfs_api
  ..ipfs_host
  ..kong
  ..moderationui
  ..mongo
  ..omaf
  ..postfix
  ..prometheus
  ..right-management
  ..solr
  ..ui
```

This folder includes all the necessary configuration files of the node.

All the following variables must be set before running the docker-compose file.

3.1.1 Mongo

``Deployme/config/mongo/.env``

These parameters define the properties needed by the MongoDB instance (e.g., admin user credentials).

```
MONGO_DB_NAME=root_db # It is the name of the DB
MONGO_ROOT_USERNAME=<not common> # It is the Admin username of the DB
MONGO_ROOT_PASSWORD=<not common> # It is the password of the Admin user
```

For example:

```
MONGO_DB_NAME=root_db
MONGO_ROOT_USERNAME=root
MONGO_ROOT_PASSWORD=*****
```

3.1.2 DAM

``Deployme/config/dam/.env``

These parameters are needed for the configuration of the DAM container and, among others, include all the properties required for the DAM to access other services of the node.

Set mongo DB parameters:

Mongo DB parameters are needed for the DAM to access the running mongo DB instance. These should be the same values as in 2.1.1.

```
MONGO_CONNECTION_URL=<not common>
MONGO_DB_NAME=<not common>
MONGO_ROOT_USERNAME=<not common>
MONGO_ROOT_PASSWORD=<not common>
```

The *MONGO_CONNECTION_URL* parameter has the following structure:

```
mongodb://MONGO_ROOT_USERNAME:MONGO_ROOT_PASSWORD@mongo:27017/MONGO_DB_NAME?authSource=admin&readPreference=primary&directConnection=true&ssl=false
```

where *MONGO_ROOT_USERNAME*, *MONGO_ROOT_PASSWORD*, *MONGO_DB_NAME* should be replaced with their actual values.

Set asset storage:

```
FILE_HOST_ENV=<not common>
```

Currently, the solution supports **local** and **S3**. **local** means that the storage used for the media assets will be based on the running machine's file system, while **S3** means that the files will be stored on Amazon.

For local storage add:

```
FILE_HOST_ENV=local
```

For external storage:

An active S3 storage must be created. Please check [S3 documentation](#). Then the following parameters must be included in the `.env``:

```
FILE_HOST_ENV=S3
S3_BUCKET_NAME=<not common>
AWS_ACCESS_KEY_ID=<not common>
AWS_SECRET_ACCESS_KEY=<not common>
AWS_REGION=<not common>
```

The domain or IP of the DAM must be configured. This is used to generate proxy preview links and deep links.

```
DAM_DOMAIN=<not common> # It is the DOMAIN of the node
```

For example:

```
DAM_DOMAIN=https://xxx.xxx.xxx.xxx:5000
```

A characteristic name for the node should be specified. This facilitates federated search as the retrieved assets are tagged with the name of the node, alongside its domain name or IP.

```
DAM_NAME=<not common>
# For example
DAM_NAME=atc-vm.gr
```

Set maximum file size limit:

Maximum file size limit can also be set:

```
SPRING_SERVLET_MULTIPART_MAX-FILE-SIZE=25MB # meaning total file size cannot exceed 25MB.
SPRING_SERVLET_MULTIPART_MAX-REQUEST-SIZE=25MB #meaning total request size for a multipart/form-
data cannot exceed 128KB.
## If not set 25MB will be the default
```

Set the Twitter configuration:

For interacting with twitter an API Key and Secret must be generated following the below steps:

Sign up for a developer account:

1. Log-in to Twitter and verify your email address. (Note that the email and phone number verification from your Twitter account may be needed to apply for a developer account, review on the Twitter help center: email address confirmation or add phone number.)
2. Click sign up at developer.twitter.com to enter your developer account name, location and use case details.
3. Review and accept the developer agreement and submit.
4. Check your email to verify your developer account. Look for an email from developer-accounts@twitter.com that has the subject line: "Verify your Twitter Developer Account" Note: the developer-accounts@twitter.com email is not available for inbound requests.
5. You should now have access to the Developer Portal to create a new App and Project with Essential access, or will need to continue your application with Elevated access.
6. If you apply for Elevated access (or Academic Research access) please continue to check your verified email for information about your application.
To check if you have a developer account, go to the developer portal dashboard to review your account status and setup.

To acquire an API Key and Secret:

Create a Twitter App: <https://developer.twitter.com/en/docs/apps> within the developer portal.

When you create your Twitter App, you will be presented with your API Key and Secret, along with a Bearer Token. Please note that these credentials are displayed only once, so make sure to save them in your password manager or somewhere secure.

Please refer to [Twitter's documentation](#) for more details.

```
TWITTER_OAUTH_CONSUMER_KEY=<not common - define the dev twitter key>  
TWITTER_OAUTH_CONSUMER_SECRET=<not common - define the dev twitter secret>
```

Set the YouTube configuration:

For interacting with YouTube Data API v3 you need to create or use an existing Google account used for the project. Follow the steps below to complete the Google project creation and configuration.

Sign into Google account:

1. From the Google's account console (<https://console.cloud.google.com/>) you should create a new project (provide project name and location).
2. From the menu on the left select APIs & services -> Enabled APIs & services.
3. From the button on the top click Enable APIS AND SERVICES and search for YouTube Data API v3. Click the card and select ENABLE, to make the YouTube DATA API active for the project. After the YouTube API is enabled, you will land at the APIs configuration page.
4. From there, navigate to the Credentials from the menu on the left.
5. From the top select create Credentials and select the API Key type of credential. This will create an API key for our YouTube enabled API which we will set to GOOGLE_API_KEY key of the .env file of the DAM. This API key will be used to perform actions to the PUBLIC available data of YouTube using the DAM integration.

GOOGLE_API_KEY = <not common - define the dev Google API key>

6. To enable DAM YouTube integration to post a video on behalf of a logged-in user to his/her YouTube channel we should also create a new Credential of type OAuth client ID. This type of credential requires to configure an OAuth consent screen, so we will proceed with this configuration first.
7. We select from the menu on the left OAuth consent screen and follow the setup steps. We choose an "External" user type and proceed with the required fields (App name, User support email and Developer contact information email.) configuration of the form.
8. In the next step to set up the SCOPES, we enable the scope from YouTube API v3 -> .../auth/youtube.upload (Manage your YouTube videos)

9. Add a test user email to the next step using the ADD USERS button and save the configuration on the last step with the summary of the OAuth consent screen setup.
10. Having the OAuth consent screen configured we can proceed with the creation of the OAuth client id credential.
11. From the create credentials button on the top select the OAuth client id credential creation, and select the “web application” type, give a web client name and a newly oauth client id credential is created.
12. After this process we will be provided with a client ID and a client Secret which will set to the DAM’s .env file to the GOOGLE_OAUTH_CLIENT_ID and GOOGLE_OAUTH_CLIENT_SECRET keys respectively.

```
GOOGLE_OAUTH_CLIENT_ID =<not common - define the dev google key>
GOOGLE_OAUTH_CLIENT_SECRET =<not common - define the dev google secret>
```

With these keys we can make requests from the DAM to the YouTube DATA API on behalf of a logged-in user to Google (only for the request that cover the requested scopes - in our case only to upload videos to YouTube).

Set the Truly Media configuration:

For setting up the connection with Truly Media platform:

1. Request a new TRULY_ORGANIZATION_API_KEY from Truly Media personnel by sending a mail to support@truly.media
2. After receiving the key include the below two env variables

```
TRULY_ORGANIZATION_API_KEY = <key received by Truly media>
TRULY_DOMAIN = <the domain of the Truly Media platform>
```

Please also notice that Truly Media platform requires a Twitter account to be present otherwise the user will not be able to sign into the platform.

Set the mail configuration:

```
MAIL_USERNAME=mail_root@DOMAIN_EMAIL_NAME # a generic user for connecting to smtp, the
DOMAIN_EMAIL_NAME must be the same as the one that will be set for postfix service
MAIL_PASSWORD=mail_password # a generic password
MAIL_HOST=postfix # the name of the postfix service
```

Keep the below values unless there is a good reason to modify them. The following parameters can be left untouched:

```
SOLR_URL=http://solr:8983/solr
IPR_URL=http://ipr-service:8081
IPFS_URL=http://ipfs_host:5001/api/v0/
TRANSCODER_URL=http://transcoder:5000
CMRR_URL=http://cmrr:8007
HATESPEECH_URL=https://services.atc.gr
NDD_URL=https://mever.iti.gr/ndd/api/v3
GRPC_URL=apis.mever.gr:443
```

```
SERVER_PORT=8888
```

```
RACU_KEY=(STXT can provide the token key upon request)  
SPRING_WEB_RESOURCES_STATIC-LOCATIONS=file:/static/
```

3.1.3 Rights-Management Components

In ``Deployme/config/right-management/`` the configuration of IPR services can be found:

``Deployme/config/right-management/mv-cicero-template-library``

mv-cicero-template-library has been configured as a git submodule which will be cloned at ``Deployme/config/right-management/mv-cicero-template-library``. The below command can be used for cloning the submodule:

```
git submodule update --init --recursive --remote
```

``Deployme/config/right-management/ipr-service/application.yml``

The bearer token that will be used for authenticating IPR service against DAM (``mv-services-bearer-tokens.mediaverse-node-backend`` property) must be set.

The process for generating the bearer token is:

1. Take the JWT_SECRET that has been defined in b)(DAM) section
2. Go to <https://jwt.io/> and put it in the VERIFY SIGNATURE
3. Add the following payload

```
{  
  "id": "6242a8e2e945ee72da204ef9",  
  "email": "ipr@mediaverse.org",  
  "username": "ipr"  
}
```

4. Keep the other default values
5. Copy the value of the generated jwt token from the encoded section

The bearer token that will be used for authenticating IPR service against Blockchain Service Provider (BCSP) must be set (``mv-services-bearer-tokens. mv-blockchain-service-provider`` property).

To generate the bearer token use:

```
docker run -it --rm mediaverse/mv-blockchain:4.0.1 npm run generatetoken
```

```
spring:  
jackson:
```

```

default-property-inclusion: NON_NULL #ObjectMapper property inclusion (do not change)

logging:
  level:
    root: INFO #define the level of logging: ERROR > WARN > INFO > DEBUG > TRACE

debug:
  disable-slc-confirmation-check: false #disable the blockchain confirmation event check

server:
  port: 8081 #port on which ipr-service is listening
  error:
    include-message: always #enable error detailed description
    include-stack-trace: 1 #enable stack-trace in errors description

mv-services-basepaths:
  mv-blockchain-service-provider: http://mv-bcsp:8082/ipr/bc #endpoint of the BCSP
  mv-slc-engine: http://mv-slc-engine:8083 #endpoint of the MV SLC Engine
  mediaverse-node-backend: http://mv-dam-api:8888 #endpoint of the MV Node Backend
  content-discovery-services-and-copyrights-negotiation: http://copyright-negotiation:3003 #endpoint of the
License Comparator

mv-services-bearer-tokens:
  mediaverse-node-backend: {defined above}
  mv-blockchain-service-provider: {defined above}
mv-services-clients-timeouts:
  connect-timeout: 60 #in seconds
  read-timeout: 60 #in seconds

```

``Deployme/config/right-management/mv-bcsp/.env``

For the development environment, paste the text below inside ``Deployme/config/right-management/mv-bcsp/.env`` and replace `DEFAULT_MV_NODE_PRIVATE_KEY` and `TOKEN_SECRET` values with yours:

```

NODE_ENV=mv-eth
DEFAULT_MV_NODE_PRIVATE_KEY=MV-NODE-PRIVATE-KEY
TOKEN_SECRET="SECRET-KEY-GENERATED-IN-PREVIOUS-STEP"

```

For the production environment, follow these steps:

1. Create an account on [INFURA](#) and generate a new API key.
2. Give a project Name and select Web3 API as network.
3. Clear BCSP configuration with this command:

```

docker run -v Deployme/config/right-management/mv-bcsp/config:/mv-blockchain-service-provider/config/ -it
--rm mediaverse/mv-blockchain:4.0.1 npm run preparedeploy

```

4. If you do not have your own wallet, you must create a new one with the command:

```
docker run -v Deployme/config/right-management/mv-bcsp/config:/mv-blockchain-service-provider/config/ -it
--rm mediaverse/mv-blockchain:4.0.1 npm run createwallet
```

NOTE: You will be asked to enter a password to encrypt the wallet. After that, the wallet address will be printed out, store it somewhere, you will need it to recharge with ETH. Meanwhile, the password you have entered will have to be written into the WALLET_PW field inside the Deployme/config/right-management/mv-bcsp/.env file (see next step). The wallet must have ETH on the chosen testnet to allow the BCSP to deploy and interact with the contracts. You can gain SepoliaETH for free by using [Infura faucet](#) using the wallet address given in the previous step. We suggest that you have at least 1 SepoliaETH to allow contracts to be correctly deployed onto the blockchain network. In addition, the admin must periodically check the availability of SepoliaETH to ensure the correct execution of transactions.

5. Paste the text below inside `Deployme/config/right-management/mv-bcsp/.env` and replace INFURA_API_KEY, TOKEN_SECRET and WALLET_PW values with yours:

```
NODE_ENV="production"
DEFAULT_NETWORK="sepolia"
INFURA_API_KEY="YOUR-INFURA-API-KEY"
TOKEN_SECRET="SECRET-KEY-GENERATED-IN-PREVIOUS-STEP"
WALLET_PW="WALLET-PASSWORD-USED-DURING-WALLET-GENERATION"
```

`Deployme/config/right-management/mv-slc-engine/application.yml`

```
logging:
  level:
    root: INFO #define the level of logging: ERROR > WARN > INFO > DEBUG > TRACE

server:
  port: 8083 #port on which mv-slc-engine is listening
  error:
    include-message: always #enable error detailed description
    include-stack-trace: 1 #enable stack-trace in errors description

debug:
  ignore-derivative-work-ownership: true # to enable nested derivative works

mv-services-clients-timeouts:
  connect-timeout: 60 #in seconds
  read-timeout: 60 #in seconds

mv-services-basepaths:
  cicero-server: http://cicero-server:6001 #endpoint of the cicero-server

slc-templates:
  library-dir: /mv-cicero-template-library #path of the SLC templates library (inside the container)
```

`Deployme/config/right-management/cicero-server/.env`


```
CICERO_PORT=6001 #port on which cicero-server is listening
CICERO_DIR=/mv-cicero-template-library #path of the SLC templates library (inside the container)
```

`Deployme/config/right-management/mv-bcsp/config/config.json` shall be empty during first run

```
{
}
```

`Deployme/config/right-management/mv-bcsp/.env`

```
RPC_CONNECTION_TIMEOUT=60000 #timeout of the RPC
```

`Deployme/config/right-management/mv-bcspeh/.env`

For the **development** environment, paste the text below:

```
NODE_ENV=mv-eth #name of the docker service container of the local blockchain deployment
IPR_SERVICE_ENDPOINT=http://ipr-service:8081/ #endpoint of the IPR Service
UPDATE_API_ENDPOINT=event/update #path of the event update API of the IPR Service
```

For the **production** environment, paste the text below:

```
NODE_ENV="production"
DEFAULT_NETWORK="sepolia"
INFURA_API_KEY="YOUR-INFURA-API-KEY"
IPR_SERVICE_ENDPOINT=http://ipr-service:8081/ #endpoint of the IPR Service
UPDATE_API_ENDPOINT=event/update #path of the event update API of the IPR Service
```

3.1.4 UI

`./config/ui/.env`

The following URLs should be defined:

URLS FORMAT = {DOMAIN_NAME}/{COMPONENT_PATH}

```
# It is the DOMAIN of the node + /dam
REACT_APP_API_URL=https://{DOMAIN_NAME}/{dam}
# It is the DOMAIN of the node + /copyright
REACT_APP_LICENSES_URL=https://{DOMAIN_NAME}/{copyright}
# It is the DOMAIN of the node + /ipr
REACT_APP_IPR_URL=https://{DOMAIN_NAME}/{ipr}
# It is the DOMAIN of the node + /ipfs
REACT_APP_FEDSE_URL=wss://{DOMAIN_NAME}/{ipfs}
REACT_APP_NODE_URL=https://{DOMAIN_NAME}
# It is the DOMAIN of the node + /template-studio
REACT_APP_SLC_STUDIO=https://{DOMAIN_NAME}/{template-studio}
```

```
# It is the maximum file size used during uploading
REACT_APP_FILE_SIZE_UPLOAD_LIMIT=1024
REACT_APP_FADER_URL=https://fader360.vrgmnts.net/users/mv-login
# It is the DOMAIN of the VRodos service
REACT_APP_VRODOS_URL=https://vrodos.iti.gr/mv-login
# It is the DOMAIN of the NERstar tool
REACT_APP_NERSTAR_URL=https://nerstar.sandec.de
```

A firebase account must be set and configured with twitter and Google authentication provider.

To set up the firebase:

1. Sign into Firebase <https://firebase.google.com/>.
(use the same Google account and project as the one created previously for the YouTube integration)
2. Click Go to console.
3. Use the project which was created previously on the YouTube integration section of the DAM. The project configuration page is open on the firebase console.
4. In the Firebase console, open the Authentication section.
5. On the Sign in method tab, enable the Twitter and the Google provider (for the Google provider no more action is needed).
6. [For twitter authentication] Add the API key and API secret from the provider's developer console to the provider configuration:
 - a. Register your app as a developer application on Twitter (see DAM's section above) and get your app's OAuth API key and API secret.
 - b. Make sure your Firebase OAuth redirect URI (e.g., my-app-12345.firebaseio.com/__/auth/handler) is set as your Authorization callback URL in your app's settings page on your Twitter app's config.
7. Click Save.
8. In the Authentication – Settings add the domain of the node as an authorized one.
9. In the Project Overview – Project Settings the auth_domain and the api key can be retrieved:
 - a. auth_domain=Project ID.firebaseio.com
 - b. api_key=Web API Key

For more details, please refer to Firebase documentation.

```
REACT_APP_FIREBASE_AUTH_DOMAIN=<not common>
REACT_APP_FIREBASE_API_KEY=<not common>
```

3.1.5 IPFS API

```
`./config/ipfs_api/.env`
```

The IPFS bootstrap node parameter helps IPFS to discover the rest of the nodes in the network.

```
IPFS_BOOTSTRAP_ADDR=<IPFS Bootstrap Node Identifier>
```

/ip4/83.149.101.53/tcp/4001/p2p/12D3KooWNRQZc9NtFVPFyG4F4jirXCk1vp4iDkgVHr3Ao9Efo1t will be used as bootstrap node for network discovery. To define it, the following shall be added in the .env file.

```
IPFS_BOOTSTRAP_ADDR=/ip4/83.149.101.53/tcp/4001/p2p/12D3KooWNRQZc9NtFVPFyG4F4jirXCk1vp4iDkgVHr3Ao9Efo1t
```

The following parameters can be left untouched:

```
IPFS_NODE_IP=ipfs_host # Container name of the IPFS Host service. Default: ipfs_host
IPFS_NODE_PORT=5001 # Port of the IPFS Host service. Default: 5001
IPFS_NODE_TIMEOUT=10
FSEARCH_RESULT_TIMEOUT=30
DAM_ADDR=http://mv-dam-api:8888
NDD_ADDR=https://mever.itigr
EXPOSE_CONTENT=True
MAX_WORKERS_NUM=4
```

3.1.6 IPFS HOST

Common for all nodes:

```
LIBP2P_FORCE_PNET=1
IPFS_PROFILE=server
IPFS_SWARM_KEY_FILE=/data/ipfs/myswarm.key
```

3.1.7 POSTFIX (Optional)

The following env variables must be in place before running the postfix container:

Variable with different value per node

```
DOMAIN_NAME=xxxxxx.yz # the email domain name
```

Variables with common values per node

```
TIMEZONE=est
MESSAGE_SIZE_LIMIT=10240000
AUTH_USER=mail_root
AUTH_PASSWORD=mail_password
DISABLE_SMTP_AUTH_ON_PORT_25=true
ENABLE_DKIM=true
DKIM_KEY_LENGTH=1024
DKIM_SELECTOR=default
```

More configuration options can be found at [https://github.com/takeyamajp/docker-postfix`](https://github.com/takeyamajp/docker-postfix)

It is very important to set DKIM keys in DNS of the domain for the mails to be delivered successfully.

3.1.8 KONG GATEWAY

The directory for the Kong Gateway configurations consists of the following sub-directories:

- **postgres** - In this sub-directory, there is the .env file that should be set for the postgres DB that KONG GW uses to store the data needed to work (default values are proposed below)

```
POSTGRES_USER=define the POSTGRES kong user | e.g., kong
POSTGRES_DB=define the POSTGRES kong DB | e.g., kong
POSTGRES_PASSWORD=define the POSTGRES kong user password | e.g., kongpass
```

- **kong-migration** - In this sub-directory, there is the .env file that should be set for the KONG GW postgres DB to bootstrap

The following parameters should be set (default values are proposed below):

```
KONG_DATABASE=postgres
KONG_PG_HOST= Should be the same as the container name of the postgres container, e.g., kong-database
KONG_PG_PASSWORD= Should be the same as the password defined as env var on postgres container, e.g., kongpass
KONG_PASSWORD= The default password used by the admin super user of the Kong Gateway. default = test
```

- **kong-gateway** - In this sub-directory, there is the .env file that should be set for the GW container to run.

The following parameters should be set (default values are proposed below):

```
#KONG_PG_HOST=<define the postgres container name that kong GW uses> <e.g., kong-database>
#KONG_PG_USER=<define the kong-postgres user> <e.g., kong>
#KONG_PG_PASSWORD=<define the POSTGRES kong user's password> | <e.g., kongpass> the same as the one set in postgres .env
```

```
KONG_DATABASE=postgres
#The following standard output/error env variables will only work on Unix environments
KONG_PROXY_ACCESS_LOG=/dev/stdout
KONG_ADMIN_ACCESS_LOG=/dev/stdout
KONG_PROXY_ERROR_LOG=/dev/stderr
KONG_ADMIN_ERROR_LOG=/dev/stderr
#The port that the Kong Admin API listens on for requests
KONG_ADMIN_LISTEN=0.0.0.0:8001
#The HTTP URL for accessing Kong Manager
KONG_ADMIN_GUI_URL=http://localhost:8002
```

- **init** - In this sub-directory, there are all the init commands that should be performed to configure KONG GW services and routes

If permission denied error occurs when curl image executes init.sh script, navigate to /kong/init/ directory and run.

```
"chmod a+x *.sh"
```

3.1.9 MODERATION UI

The following URL should be defined:

URL FORMAT = {DOMAIN_NAME}/{COMPONENT_PATH}

```
# It is the DOMAIN of the node + /dam/
REACT_APP_API_URL=http://{DOMAIN_NAME}/{dam}/
```

3.1.10 FADER

The following should be defined:

```
PHX_SERVER="true"
FADER360_BACKEND_VERSION=<Version number is imp to specify the image from which the container should
run>
FADER360_BACKEND_VERSION_NUMBER=<Version number is imp because of the version dependent assets and
preview assets path inside the container>
# eg. FADER360_BACKEND_DATABASE_URL=ecto://postgres:postgres@vragments-db/faderomafdb
FADER360_BACKEND_DATABASE_URL=<ecto          database          url,          see
https://hexdocs.pm/ecto/Ecto.Repo.html#module-urls>
# could be any 64 long string
FADER360_BACKEND_SECRET_KEY_BASE=<encryption      secret,      can      be      generated      via
https://hexdocs.pm/phoenix/Mix.Tasks.Phx.Gen.Secret.html>
FADER360_BACKEND_SCHEME=<url scheme, e.g., https>
FADER360_BACKEND_HOST=<url domain, e.g., fader360.vrgmnts.net>
FADER360_BACKEND_PORT=<url port, e.g., 443>
FADER360_BACKEND_CONTAINER_PORT=<application listening port inside container, e.g.: 17000>
```

3.1.11 OMAF

The following should be defined:

```
PHX_SERVER="true"
MV_OMAF_VERSION=<Version number is important because of version dependent asset path inside container>
MV_OMAF_DATABASE_URL=<ecto database url, see https://hexdocs.pm/ecto/Ecto.Repo.html#module-urls>
MV_OMAF_SECRET_KEY_BASE=<encryption      secret,      can      be      generated      via
https://hexdocs.pm/phoenix/Mix.Tasks.Phx.Gen.Secret.html>
MV_OMAF_SCHEME=<url scheme, e.g., https>
MV_OMAF_HOST=<url domain, e.g., mv_omaf.vrgmnts.net>
MV_OMAF_PORT=<url port, e.g., 443>
MV_OMAF_CONTAINER_PORT=<application listening port inside container, e.g.: 16000>
MV_OMAF_TRANSCODED_ASSETS_PATH=<version dependent asset path inside container, must be:
'/root/mv_omaf_release/lib/mv_omaf_web-VERSION/priv/static/assets/mv_omaf_transcoded_files', VERSION
must match image version, whole string must match mounted volume>
```

3.1.12 Prometheus and Grafana (Optional)

For monitoring the node, prometheus and grafana tools are supported.

Firstly, the Prometheus component must be set:

``Deployme/config/prometheus/prometheus.yml`` must be included. The default settings are fine, and they can be customized as well, eg. the scraping interval.

```
global:
  external_labels:
    monitor: devops_monitor
  scrape_interval: 5s
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets:
        - "localhost:9090"

# - job_name: node_exporter
#   static_configs:
#     - targets:
#       - "node_exporter:9100"

- job_name: kong
  static_configs:
    - targets:
      - "kong:8001"
```

Prometheus can be configured to listen to the DAM traffic.

The Prometheus plugin must be set by ordering:

```
curl -X POST http://localhost:8001/routes/dam-route/plugins \
  --data "name=prometheus" \
  --data "config.per_consumer=false"
```

Reference: ``https://docs.konghq.com/hub/kong-inc/prometheus/configuration/examples/``

The prometheus interface will be available at:

``http://<IP>:9090/targets?search=``

After that the grafana component must be set to add Prometheus as a data source and the official Kong Dashboard to be imported.

Grafana UI can be accessed at:

``http://<IP>:9000/``

The default username and password for Grafana is admin / admin.

The dashboard json can be downloaded here:

<https://grafana.com/api/dashboards/7424/revisions/7/download>

And the process of importing it to the grafana service is the below:

1. Open your Grafana portal and go to the option of importing a dashboard.
2. Go to the “Upload JSON file” button, select the kong-official_rev7.json which you got from the url above.
3. Configure the fields according to your preferences and click on Import.

The dashboard will be ready.

3.2 Docker Execution


Run the application, from the root level order:

```
docker-compose pull # pull the images
docker-compose up -d # run the docker-compose
```

4 User Manual

This section provides an overview of the user experience and implemented features through screenshots. This overview can serve as a user guide for the MediaVerse node.

Figure 5 depicts the Login/Create account page. In case a user already has an account, he/she is logging in with his/her credentials (email, password) otherwise the option to create an account should be selected.




Login
Email

Password

[Create account](#)
[Terms and Conditions](#)
[Privacy policy](#)

Figure 5: Login webpage

The “create account” option leads to the corresponding page. Users must fill in the required details (email, username, etc.), accept the terms and conditions and the privacy policy after careful reading them, and then create their account (see details in Figure 6).



Create account
Email

Username

Firstname

Lastname

Date of Birth

Password

Confirm password

☒ I agree to the terms and conditions
☒ I agree to the privacy policy

[Already have an account? Login](#)

Figure 6: Create account page

Once a user’s account is created the homepage page appears (Figure 7). There, one can see the main menu on the left with the Dashboard page selected by default and a secondary menu with a help button, the language option, and the profile details on the upper right of the page. Any notifications related to the user will appear on the dashboard page, and there is also a possibility to send invitations to other users. Social media login is available through this page and the analytics dashboard tool (Truly Media) is also accessible with the available social media (Twitter, YouTube).

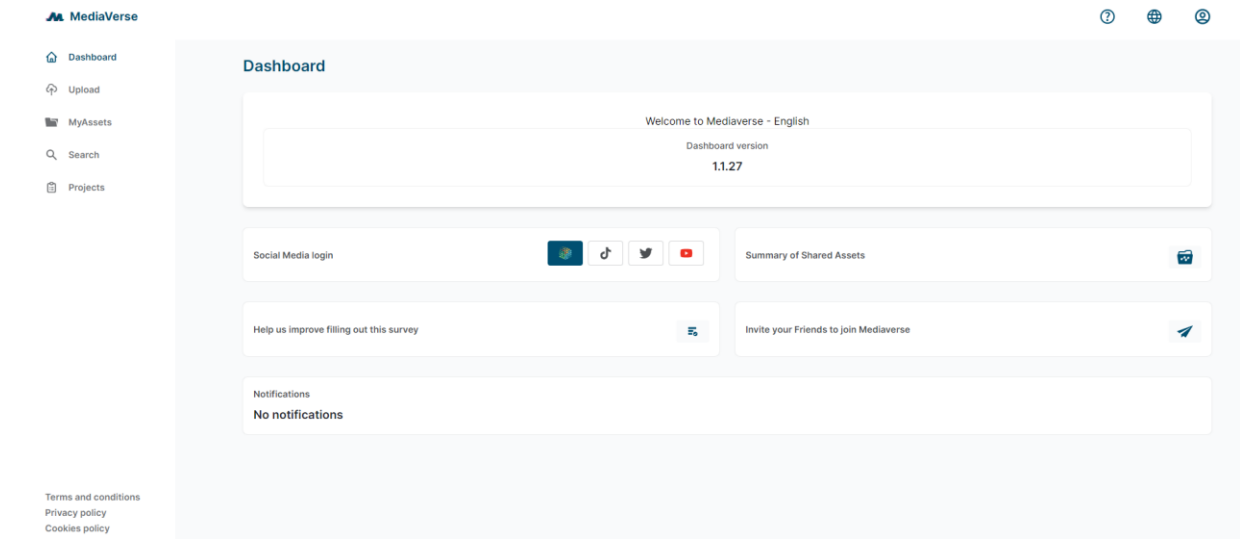


Figure 7: Dashboard page

The next item in the main menu is the Upload functionality (Figure 8). On this page, content guidelines are available to users, which define what kind of content is not allowed to be uploaded in MediaVerse. The **Drop your file here** button is the main functionality of this page where one can upload the desired content (audio, images, videos, and 3D videos).

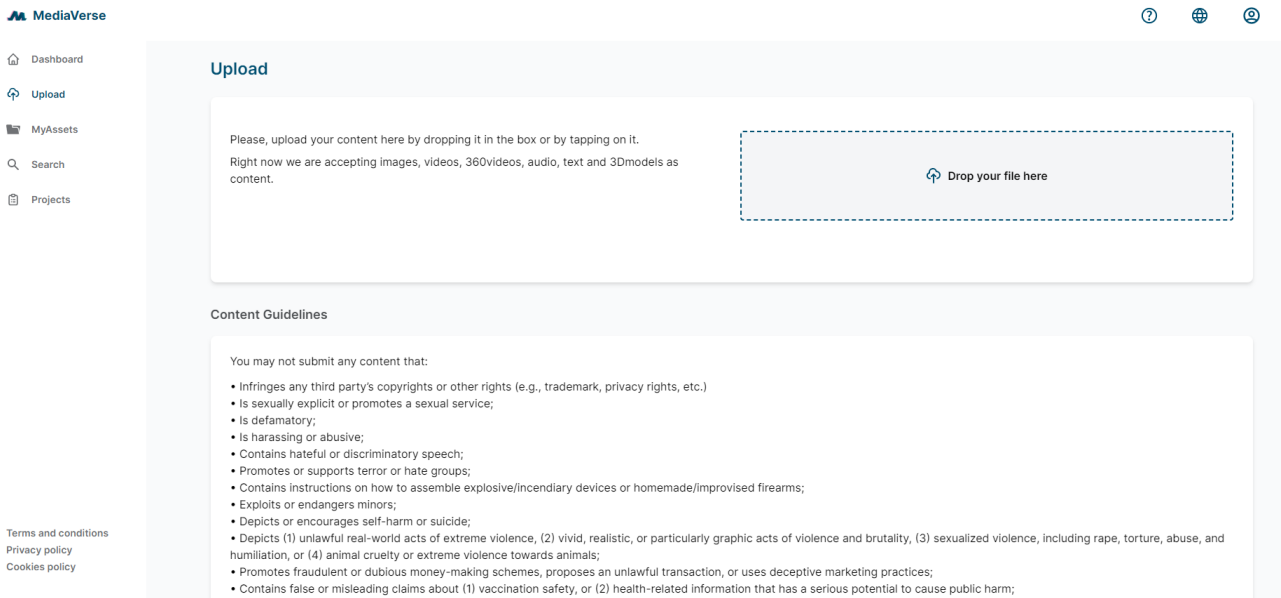


Figure 8: Upload page

Once an item is about to be uploaded a pop-up page opens to fill the details of an asset (Figure 9).

Fill in your asset metadata for photo-1531804055935-76f44d7c3621.jpg

Description:

photo-1531804055935-76f44d7c3621.jpg

Original Filename

photo-1531804055935-76f44d7c3621.jpg

Content Type

image/jpeg

Content Creation Date

26/05/2023

Price (EUR)

cost

Flags

☐ Satire

Available Languages

Add...

Labels

Add...

I WILL DO THIS LATER

Done

Figure 9: Fill details when uploading an asset

In the “MyAssets” item in the main menu, all the available content (e.g., previous projects, videos, etc.) of the specific user will be displayed along with options of filtering the displayed results (Figure 10).

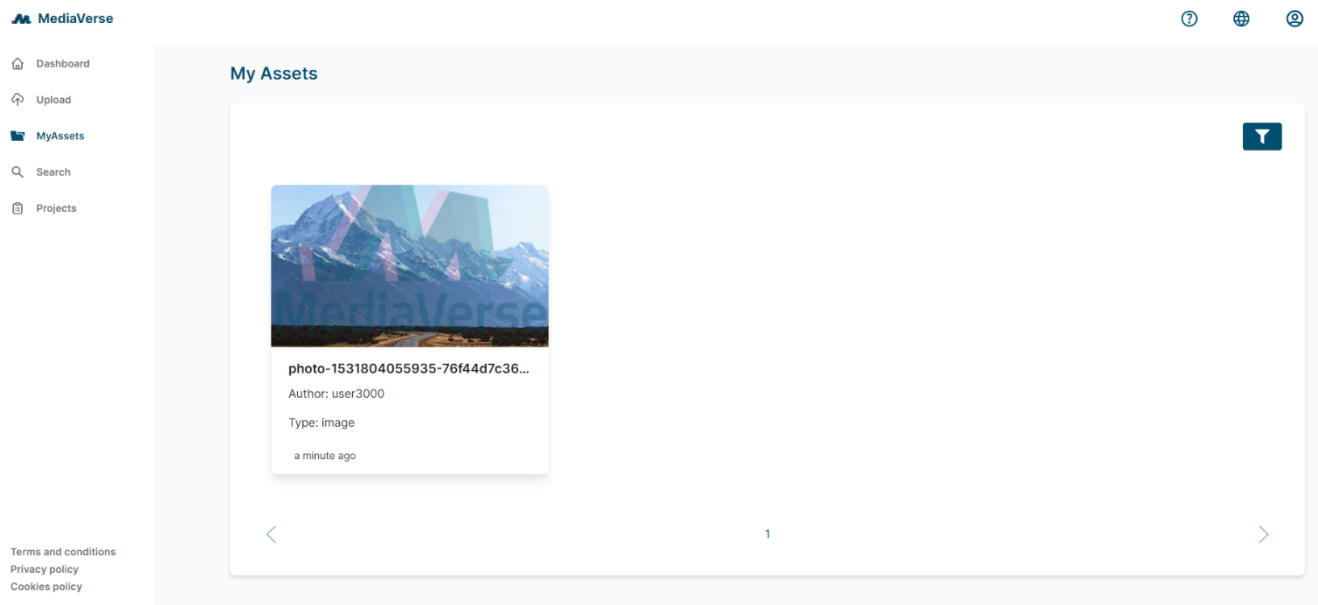


Figure 10: My Assets page

When an asset is selected a page will be displayed (Figure 11), showing all the available details of a specific asset (e.g., description, content creation date, etc.).

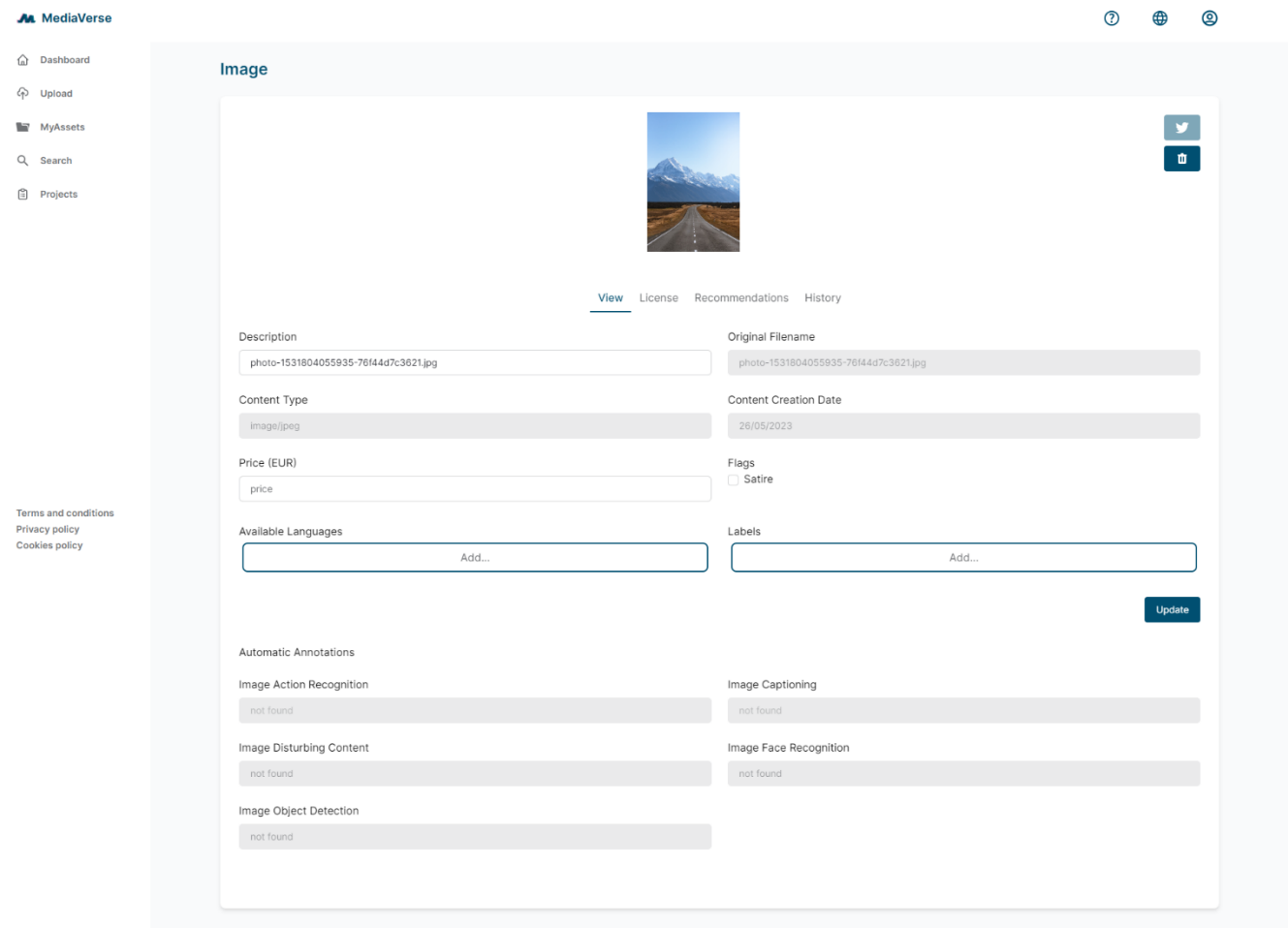


Figure 11: Assets details

Figure 12 provides an example of asset license registration. For a given asset uploaded by a user, a specific license from the supported ones can be selected and the corresponding transaction is committed in the Ethereum network.

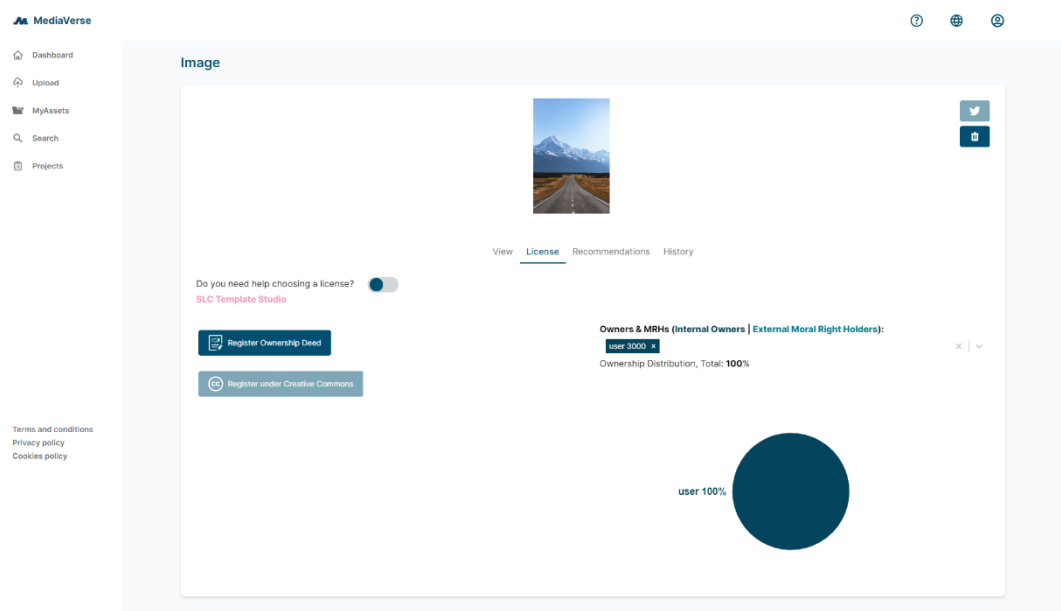


Figure 12: License registration for an asset

In the recommendations page similar content will be provided to the user considering previous selections (Figure 13).

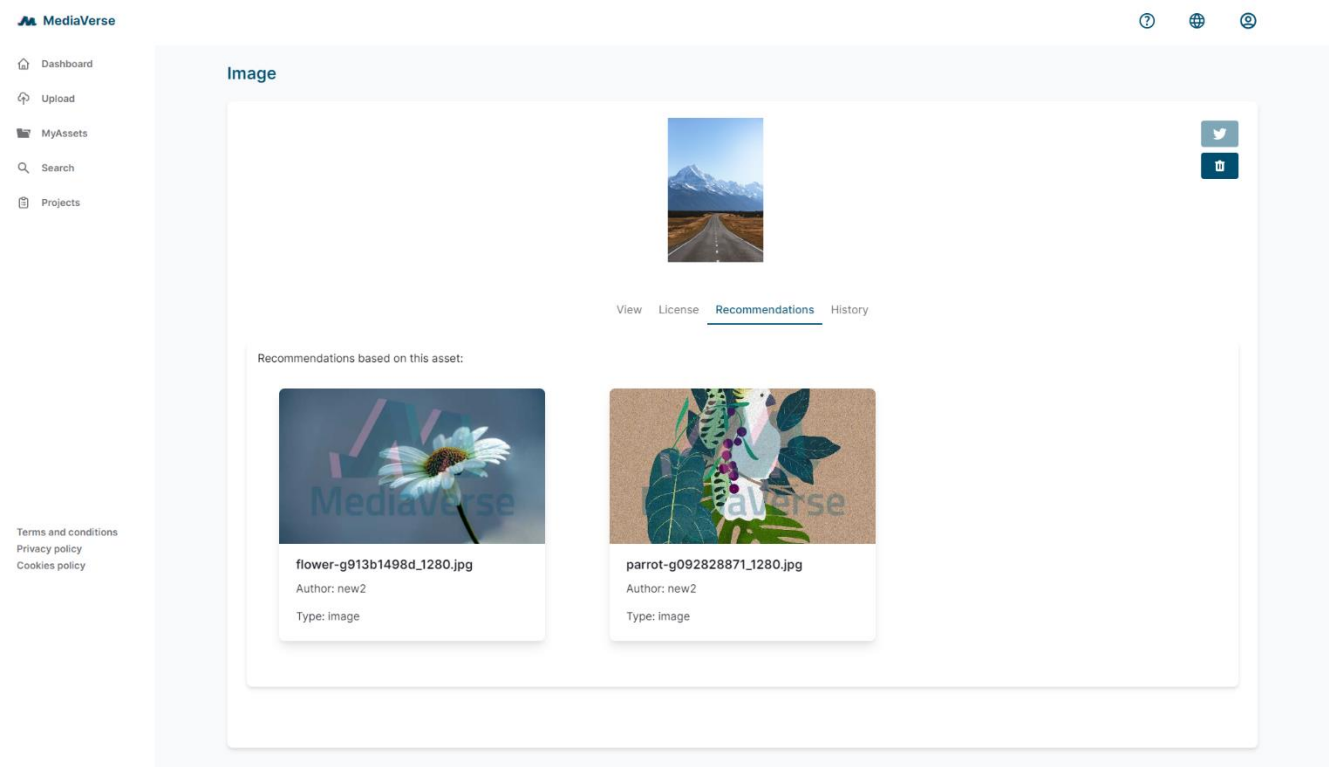


Figure 13: Recommendations page

The history page (Figure 14) presents the history of an asset i.e., who has used it before and in which projects was it used.

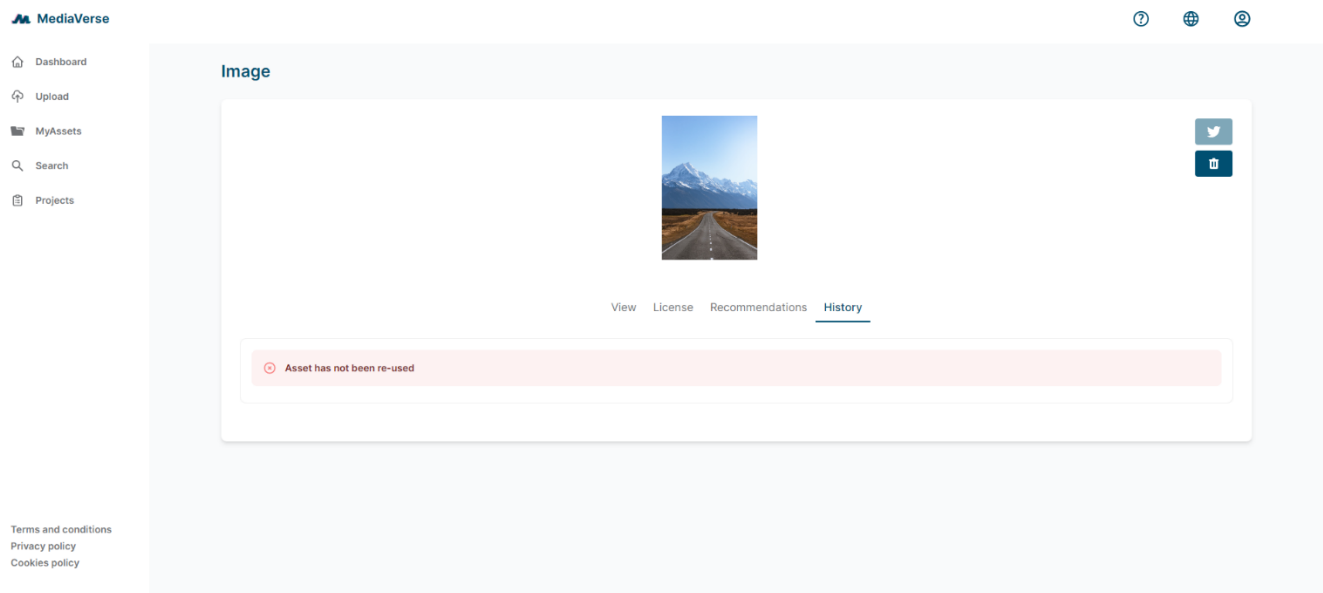


Figure 14: History page

In the Search page (Figure 15) there are three search options. The recommendations which return recommendations of assets similar to someone's interests, the local/federated which returns results from either the local MV node or other available nodes. Finally, the external search which returns any related result found in WikiMedia Commons⁸. There are also some filter options that can be applied on the searches, such as to return videos or photos and the number of assets to be displayed per page.

⁸ https://commons.wikimedia.org/wiki/Main_Page

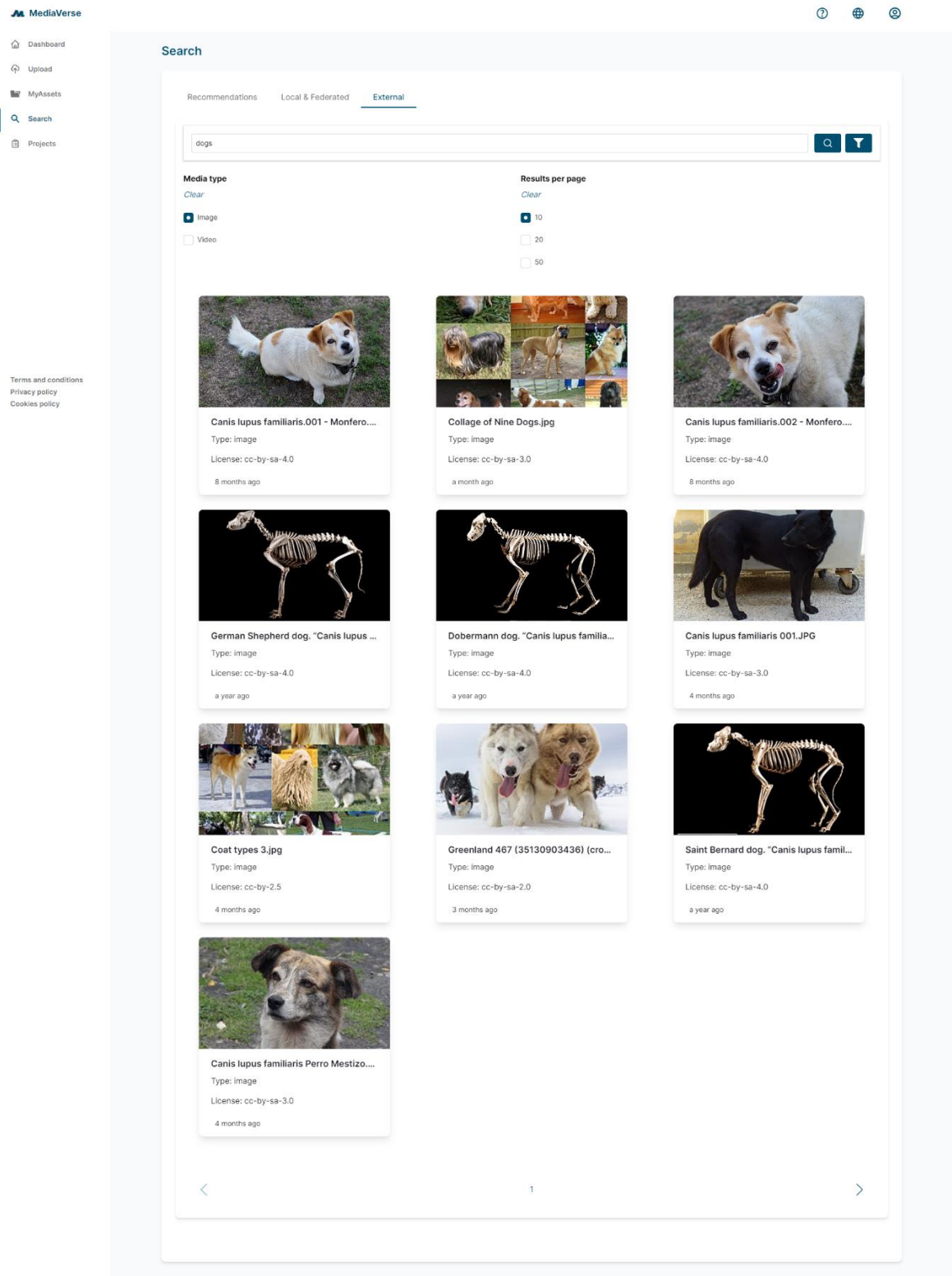


Figure 15: Search page

The final item in the main menu is the Projects tab (Figure 16), which contains all the projects that the specific user is involved in (either as an owner or as participant). These projects are collections of assets which can be used for authoring procedure. With the integration of authoring tools in the MV platform (Fader and VRodos),

projects created in MediaVerse and the associated assets, would be available within these authoring tools in order to create derivative works.

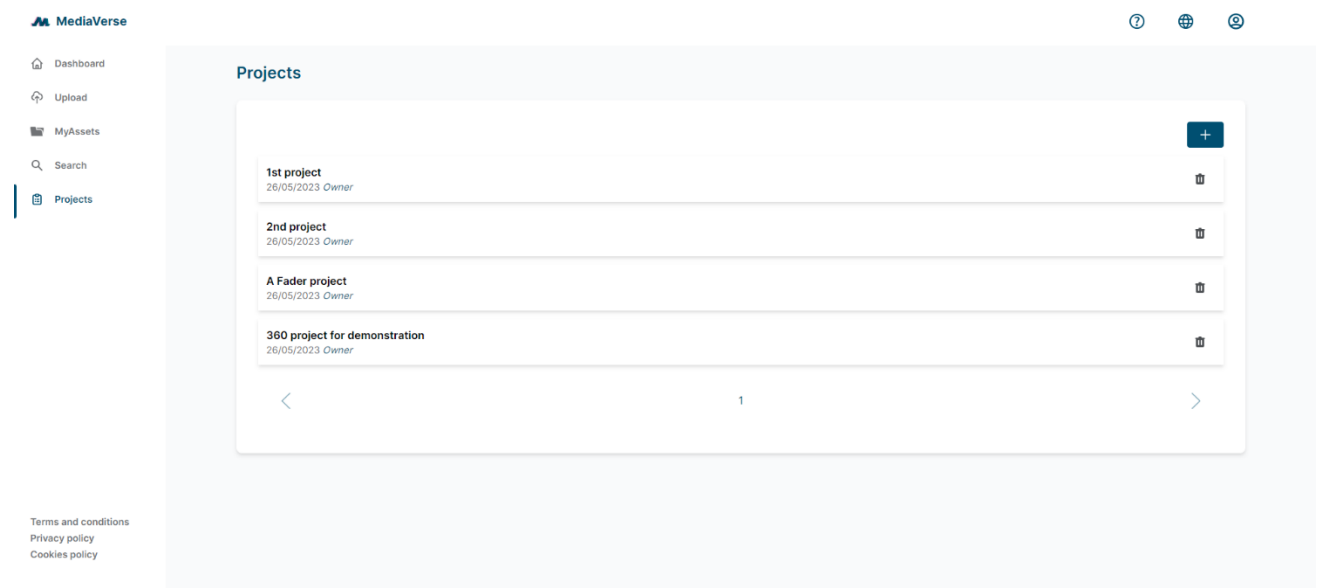


Figure 16: Projects page

When a MV user selects a project, the details appear in a new page (Figure 17). Therein, information about the project can be found like the owner or other participants, date created, add users as owners or contributors, assets of the project, license, etc. Last, the user can select the options of the authoring tools (VRodos, Fader) to further edit the project.

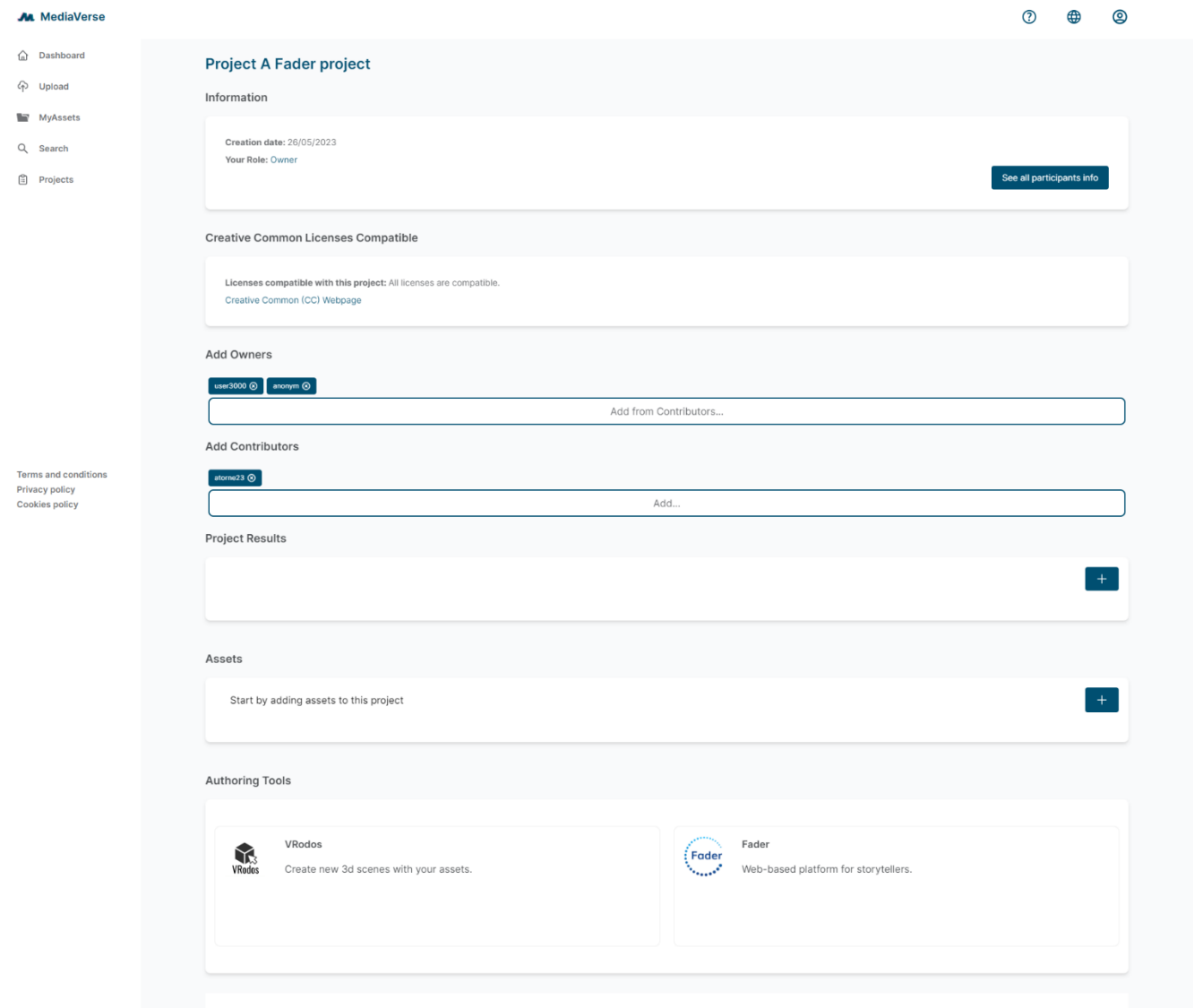


Figure 17: Project details page

The links to the external tools VRodos and Fader leads to the corresponding login pages of the external tools (Figures 18-19).

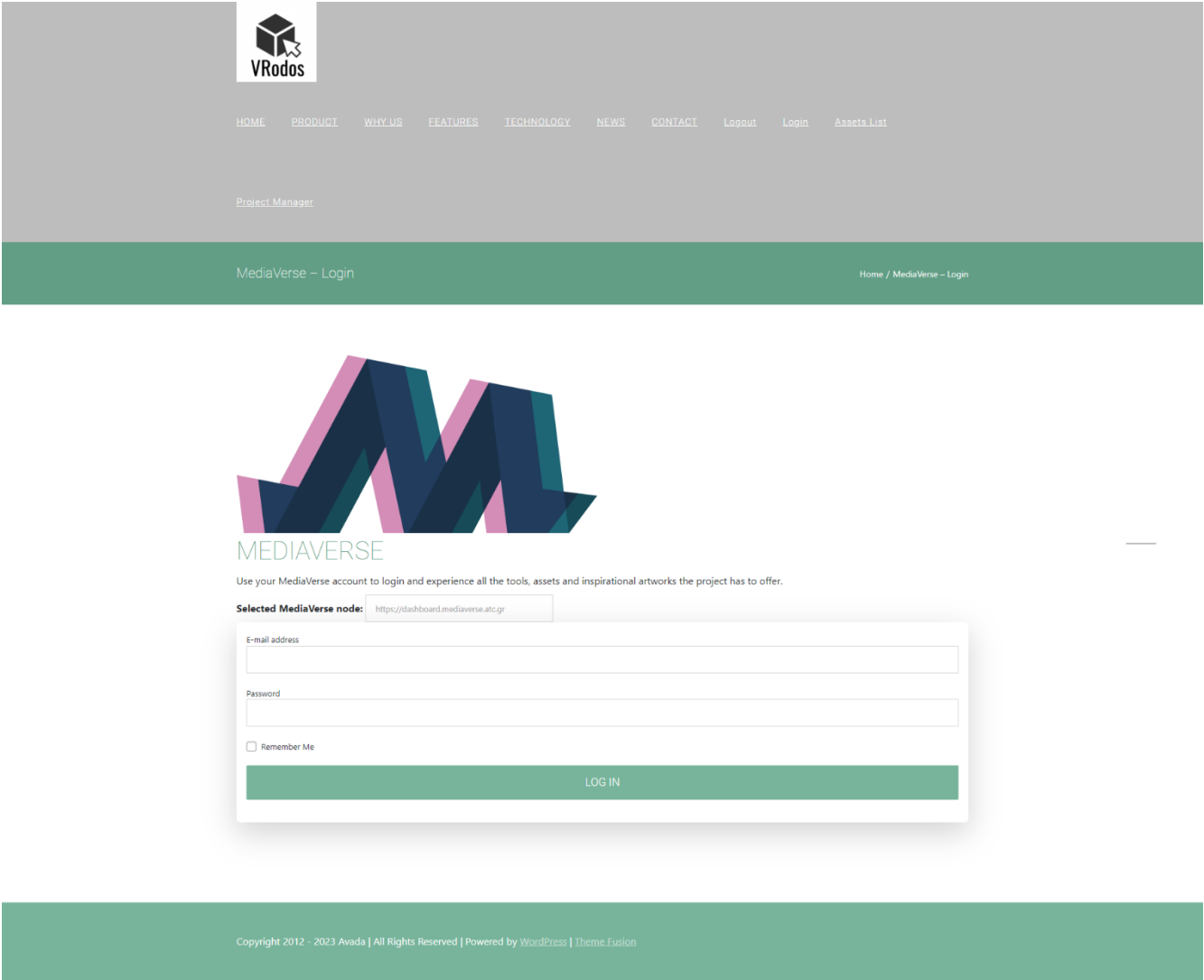


Figure 18: VRodos login webpage

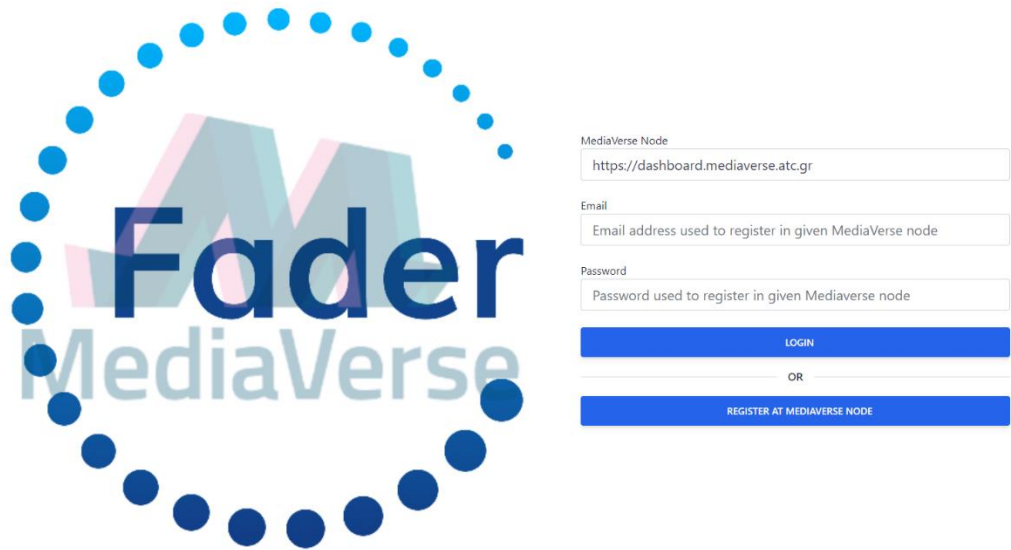


Figure 19: Fader login webpage

In the secondary menu located in the upper right of the page (Figure 20), from left to right one can see the help functionality (pop up window that explains the main menu options), the language selection option (currently Catalan, English, and Spanish) and the profile page option.

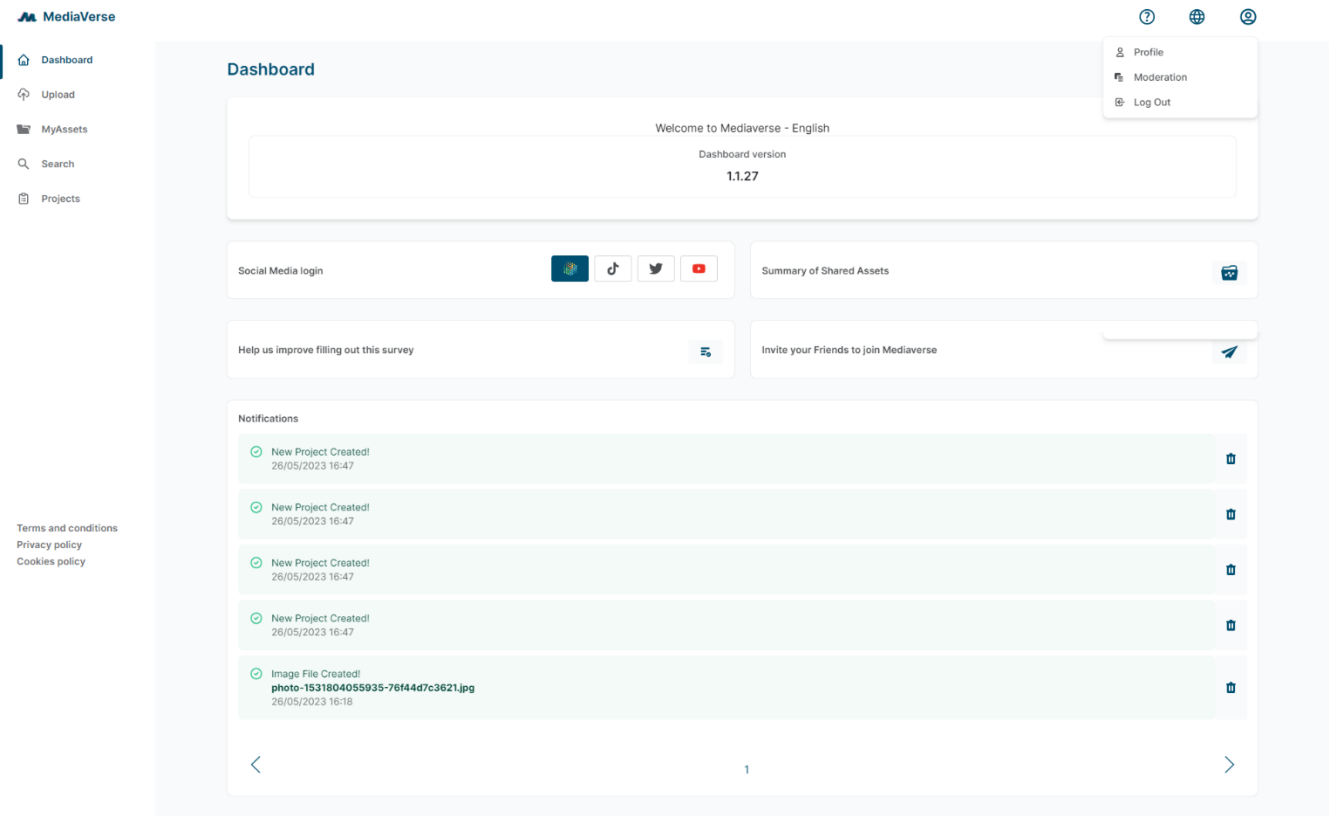


Figure 20: Secondary menu

On the profile page (Figure 21), the relevant information with respect to the profile of the user will be displayed (e.g., username, date of birth, billing information, close account, etc.). In addition, this section can be used to update the user profile and set a registered office address which is mandatory from a legal perspective for the user to register assets.

The screenshot displays the 'Profile' page of the MediaVerse application. The page is divided into several sections:

- User Info:** Contains fields for Email (user2000@gmail.com), Username (user2000), Date of Birth (02/01/1988), First name (user), Last name (2000), and Registered Office Address (Kallistratos 10, Athens, Greece). An 'Update' button is at the bottom right.
- Billing information / Wallet:** Contains fields for Blockchain User Address (0x1234567890123456789012345678901234567890), PayPal Seller Token (paypal123456), and Preferred Fiat Currency (USD). An 'Update' button is at the bottom right.
- Settings:** Includes a checkbox for 'Show subtitles' and a 'Content Moderation' button.
- Professional Role:** Contains fields for Title (journalist), Company (VRS), Description (My first image), and Skills (Select...). An 'Update' button is at the bottom right.
- Close account:** A section with a warning message 'Close current account. This action can't be reverted.' and a 'Close' button.

The left sidebar contains navigation links: Dashboard, Upload, MyAssets, Search, Projects, Terms and conditions, Privacy policy, and Cookies policy.

Figure 21: Profile details

Finally, the Moderation option (Figure 22) gives the ability to the user to flag any content as Not Safe for Work (NSFW) or as Disturbing Image. Then, the content moderator will decide on the content and take the appropriate measures as flag an image as NSFW content.

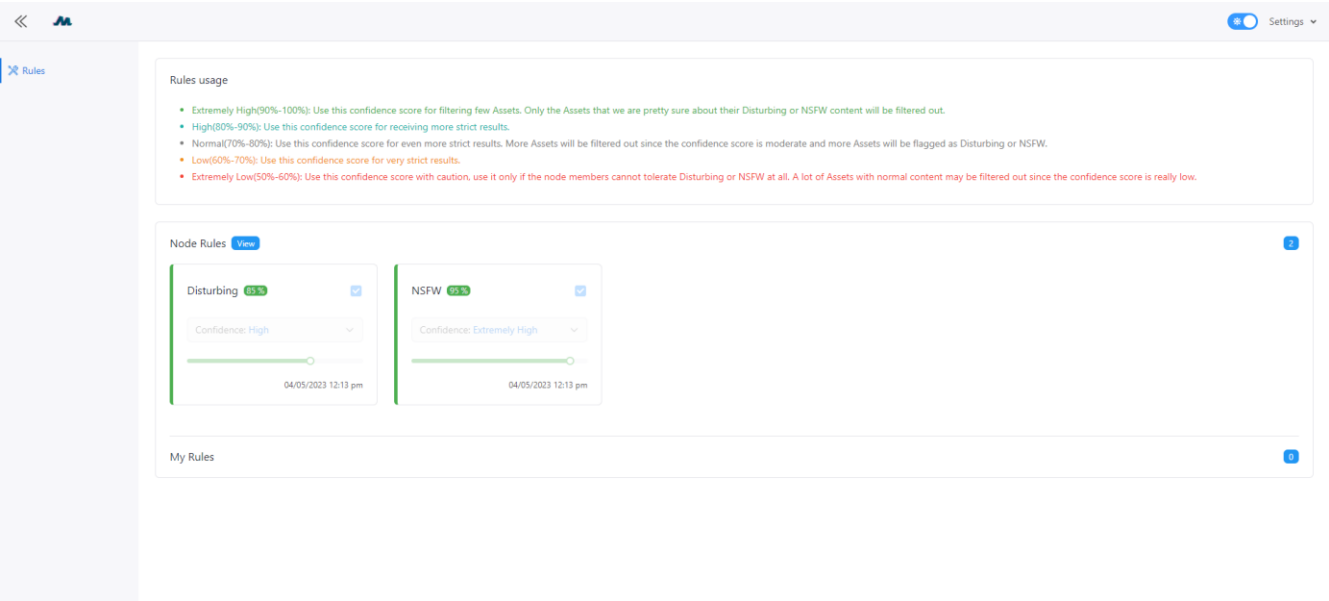


Figure 22: Moderator page

5 Testing

This section provides an overview of the testing strategy that has been designed and followed for verifying the project's functionalities. The test cases that have been created along with all the testing plan specifications are presented.

5.1 Testing Plan

The specifications of the testing plan that has been followed can be seen in Table 2.

Table 2: Overview of the testing plan

SPECIFICATION ITEM	WHAT WE FOLLOWED
Scope	Verify functionalities and requirements
Testing type	System test
Environment	3 live nodes
Tools	Manual testing
Defect management	Gitlab issues
Risks	Hardware limitations

A manual system testing against three live nodes (see 1.3 section) has been decided for verifying the functionalities and the requirements of the project. Any defects that may arise are inserted as tickets in the Gitlab platform that is used by the partners for tracking the development process. Risks that have been taken into account are regarding the hardware limitations of each node, e.g., when storage is limited in one node, test cases that depend on uploading huge files at some point will fail and may also lead to total system failure.

5.2 Test Cases

The test cases that have been designed are separated in 13 categories, which reflect the most important features of the platform. The focus of the testing procedure is on the Asset management of the platform and the interaction of the Assets with the IPR, the transcoding and the authoring services. Finally, as any testing procedure the test cases are dynamic. Below the basic test cases are presented but more details can also be found in a [live document](#) shaped by all partners.

Asset uploading

1. Images
 - a. Upload a jpg normal size ~5MB
 - b. Upload a png normal size
 - c. Upload a TIFF normal size
 - d. Upload a BMP normal size
 - e. Upload a SVG normal size
 - f. Upload a GIF normal size
 - g. Upload a WebP normal size
 - h. Upload a large image ~20MB
2. Videos
 - a. Upload a mp4 normal size ~50MB - ~100MB

- b. Upload an avi normal size
 - c. Upload a 3GP normal size
 - d. Upload a flv normal size
 - e. Upload a mkv normal size
 - f. Upload a webm normal size
 - g. Upload a huge video ~1GB
- 3. 3D Model
 - a. Upload an OBJ normal size ~200MB
 - b. Upload a FBX normal size
 - c. Upload a DAE normal size
 - d. Upload a STL normal size
 - e. Upload a huge 3D ~1GB
- 4. Audio
 - a. Upload an AIFF normal size ~5MB
 - b. Upload a WAV normal size
 - c. Upload a FLAC normal size
 - d. Upload a mp3 normal size
 - e. Upload a huge audio ~100MB

Asset modification

- 1. Update metadata
 - a. Update description
 - b. Update Languages
 - c. Update Labels
 - d. Update Satire flag
 - e. Update price
- 2. Delete
 - a. Delete asset (one owner)
 - b. Delete asset with multiple owners

Asset registration

- 1. Only own owner
 - a. CL license (price is set)
 - b. CC licenses registration
 - i. CC_BY
 - ii. CC_BY_SA
 - iii. CC_BY_NC
 - iv. CC_BY_NC_SA
 - v. CC_BY_ND
 - vi. CC_BY_NC_ND
 - vii. CC_0
- 2. Multiple owners
 - a. Set 1 internal owner
 - b. Set 1 external owner
 - c. Set 1 internal and one external owner

- d. CC licenses registration
 - i. CC_BY
 - ii. CC_BY_SA
 - iii. CC_BY_NC
 - iv. CC_BY_NC_SA
 - v. CC_BY_ND
 - vi. CC_BY_NC_ND
 - vii. CC_0
- e. Approve proposal (notification to internal owner)

Asset searching

- 1. Local search
 - a. Search by nsfw flag
 - b. Search by disturbing flag
 - c. Search by nsfw and disturbing
 - d. Search by license
 - e. Search by media type
 - f. Search by dates
 - g. Search by free-text query
- 2. Federated search
 - a. Search by nsfw
 - b. Search by disturbing
 - c. Search by nsfw and disturbing
 - d. Search by license
 - e. Search by media type
 - f. Search by dates
 - g. Search by free-text query

Asset acquisition

- 1. Import
 - a. Import a small asset of your own
 - b. Import a small asset of a creator, no other owner, same node
 - c. Import a small asset of a creator, multiple other owners, same node
 - d. Import a small asset of a creator, no other owner, different node
 - e. Import a small asset of a creator, multiple other owners, different node
 - f. Import a huge video of same node
 - g. Import a huge video of different node
 - h. Import an asset from a node A to node B, import it again for different owner in node B, import it again for different owner in node C
- 2. CL SLC flow (check what happened with SLC after importing)

Asset annotations

- 1. Image Action Recognition
- 2. Image Captioning
- 3. Image Disturbing Content

4. Image NSFW Content
5. Image Object Detection
6. Video Action Recognition
7. Video Disturbing Content
8. Video NSFW Content
9. Video Object Detection

Projects

1. Create project
2. Delete project
3. Add contributors
4. Add assets
5. Ask contributors to add content in the same project
6. Export project to an authoring tool
7. Add project results from authoring tool
8. Register rights of project result (i.e., Derivative Work workflow)

Subtitles

1. Add subtitles manually
2. Update subtitles with NERstar editor
3. Update again subtitle with NERstar editor
4. Use automatic subtitles from RACU

User profile

1. Update username
2. Update date of birth
3. Update first name
4. Update last name
5. Update address
6. Update paypal token
7. Update preferred fiat
8. Update settings
9. Update professional role
10. Close account (this will permanently delete the account)

Login / Logout / Forgot pwd

Registration of a new user

Social media & external tools

1. Connect Twitter
 - a. Share Image
 - b. Share Video
2. Connect Youtube
 - a. Share Video
3. Truly media

4. Logouts from social media

Notifications

1. Delete a notification
2. Show notifications (all types)
3. Approve notification

6 Conclusion

MediaVerse has successfully delivered on its mission to provide a decentralized platform for media asset management. Through research and development efforts, project partners have brought to life a fully functional platform that addresses user requirements and incorporates state of the art technologies. The final release of MediaVerse marks a significant milestone in the way media assets are managed, setting the stage for a more open, efficient and decentralized media ecosystem.



MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252.
The content of this document is © the author(s). For further information, visit mediaverse-project.eu.