MediaVerse

A universe of media assets
and co-creation opportunities

D5.3

VR Collaboration, Social Media Analytics, and Immersive Storytelling - Authoring Tools 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	Immersive Storytelling toolset
Deliverable number	D5.3
Deliverable version	V1.0
Previous version(s)	D5.2
Contractual Date of delivery	31.08.2022
Actual Date of delivery	31.08.2022
Nature of deliverable	Other
Dissemination level	Public
Partner Responsible	CERTH
Author(s)	Dimitrios Ververidis (CERTH), Quiros Fernandez, Enrique (ATOS), Spyros Papafragkos (ATC), Stephan Gensch (VRAG) , Manos Scinas (CERTH), Manios Krasanakis (CERTH)
Reviewer(s)	Violeta Vasileva (ARTSHARE), Stephan Gensch (VRAG), Symeon (Akis) Papadopoulos (CERTH)
EC Project Officer	Luis Eduardo Martinez Lafuente

Abstract	This deliverable reports on the developments in relation to a) the support of 3D models uploading and previewing in the main platform, b) the VR authoring tools as external services, and c) the Social Analytics Engine.

Keywords	VR Authoring Tools, Immersive Storytelling, VR Sandbox, Analytics

Copyright

© Copyright 2021 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	10/02/2022	Dimitrios Ververidis (CERTH)	First Draft Table of Content, including Sections 2, 3, 4 and 6
V0.2	25/2/2022	Quiros Fernandez, Enrique (ATOS)	Section 5
V0.3	1/3/2022	Dimitrios Ververidis (CERTH) Efstratios Tzoanos (ATC)	Editing and review, Add Table of Contents, Figures and Tables (CERTH) Initial section for Social Analytics (ATC)
V0.4	25/3/2022	Dimitrios Ververidis (CERTH) Efstratios Tzoanos (ATC)	Intermediate version addressing comments by Symeon Papadopoulos (CERTH)
V0.5	08/08/2022	Stephan Gensch (VRAG)	Contributing to immersive storytelling toolset
V0.6	23/08/2022	Manos Schinas, Symeon Papadopoulos, Manios Krasanakis (CERTH)	Review and comments
V0.7	24/08/2022	Dimitrios Ververidis (CERTH)	Final version addressing comments
V0.8	27/08/2022	Quiros Fernandez, Enrique (ATOS)	Update content on “Support of the uploading of 3D Models and 360 videos in the MediaVerse Node”
V0.9	30/8/2022	Dimitrios Ververidis (CERTH)	Several refinements
V1.0	31/08/2022	Symeon Papadopoulos (CERTH), Evangelia Kartsounidou (CERTH	Final review and QA

Glossary

ABBREVIATION	MEANING
AP	Average Precision
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
DAM	Digital Asset Management
FG	Focus Group
GDPR	General Data Protection Regulation
GUI	Graphic User Interface
HMD	Head-mounted Display
ML	Machine Learning
MV	MediaVerse
NAF	Networked-Aframe
NAS	Neural Architecture Search
NER	Named Entity Recognition
NLP	Natural Language Processing
SCRFD	Sample and Computation Redistribution for Efficient Face Detection
UC3	Use Case 3
UGC	User Generated Content
VR	Virtual Reality
WP	Work Package
XR	eXtended Reality

Table of Contents

Index of Figures

Index of Tables

Executive Summary

In year 2, the developments that took place in tasks T5.3, T5.4, and T5.5 regard the implementation and deployment of the respective three services as well as their integration to the MediaVerse nodes. These three tasks deliver external services that are interconnected to each MediaVerse node through a REST API. Namely, these services are for creating VR experiences such as 360 based ones for T5.3 or native 3D ones for T5.5, and enriching the existing content through social media platforms as well as monitoring the overall performance of the MV node platform through analytics for T5.4.

For T5.3, we leverage Fader, a SaaS by VRAG that allows for interactive 360-degree immersive storytelling. It uses uploaded media, displaying and arranging it in 3D WebGL scenes. It employs the MediaVerse Digital Asset Management (DAM) interface to retrieve and process media files for use in Fader. It can also use improved 360-degree video transcoding under WP3, T3.3. Accessibility features are enhanced by using available subtitles for videos that are generated by the accessibility tools. Fader can be deployed either standalone or on premise within the same architecture of a node. Both the Fader backend and the front-end (player and editor) are open-sourced, so maximum flexibility for future deployment and use cases are given. We have also given emphasis on the anonymization of content such as the blurring of human faces in footage content.

For T5.4, we have made available a Social Analytics engine based on the existing commercial solutions by ATC, Truly Media and TruthNest, by extending their functionality and by integrating them to the MediaVerse node, either as extension services or via the consumable API.

For T5.5, we have used VRodos by CERTH as a basis for authoring 3D experiences through a web interface. We have enhanced VRodos with the functionality of creating virtual productions from remotely located actors, musicians or journalists in real-time using Web Real Time Communication protocol (WebRTC), and WebGL for rendering the background scenery 3D models. This kind of new media combines 2D content for actors with 3D content for scenery. The final experiences are a) a URL link in VRodos server that is a 3D real-time experience for a web browser; and b) a virtual production movie that was recorded during the immersion of actors, and it is given back to a MediaVerse node as new footage content.

1 Introduction

This deliverable describes the latest implementations on services that can retrieve content from MediaVerse nodes, process it, and return it in a more attractive format. These services are tools related to 3D and 360 content creation, and tools related to content enrichment with information extracted from social media. We present the latest developments for the support of creating 360 media based experiences out of 360 footage content (T5.3), the support of a unified service for enhancing MediaVerse experiences with new content extracted from social media platforms (T5.4), and the creation of 3D experiences out of 3D models found in MediaVerse nodes (T5.5).

1.1 Relation with Other Activities in MediaVerse

Social Analytics functionality developments rely on the Conceptual Architecture Design of WP2, which has been described in D2.2 - Conceptual Design of the MediaVerse Framework[1] and the integration plans from WP6 that have defined the mechanism to integrate the functionality in a MediaVerse Node ecosystem. The WP2 and WP7 will also provide the user feedback to define the specifications for the fine-tuning of Social Analytics features.

VRodos and Fader are implementations towards VR experiences that use a variety of MediaVerse services, such as media processing and annotation, digital asset management, or accessibility tools, to name a few (see Figure 1). WP2 gives a good indication on the envisioned architecture that serves as a foundation to integrate the VRodos and Fader authoring tools that this deliverable describes in technical terms. The deliverables of WP5 describe these services, while WP7 will provide feedback in order to refine the developments on these tools.

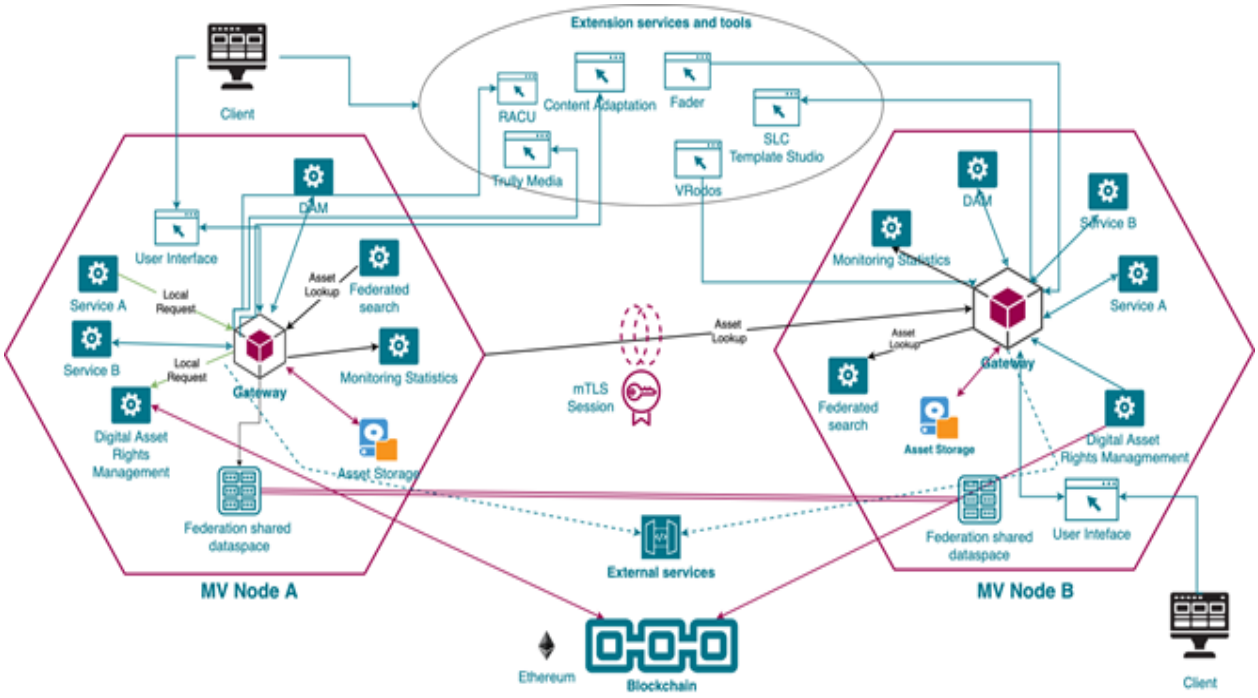

Figure 1: The position of VRodos and Fader with respect to MediaVerse architecture.

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

WP2

The use cases and requirements engineering in WP2 are a crucial part to determine what users expect from the authoring tools. Section 2 of this deliverable deals with the comparison of the three main types of XR technologies and derives comparable templates. In addition, WP2 defined a general architecture design process that determined the integration of the authoring tools into the MediaVerse architecture. The authoring tools presented in this deliverable due to their third-party nature and have many challenges to overcome, from authentication to media management. Such challenges are both technical and user oriented. Technical challenges refer to the way that the authoring tools reference media files, deal with (temporary) local copies for processing, and provide back to a MediaVerse node the resulting experiences. User oriented challenges are related to the engagement of people in new types of social media such as VR in which the technology barrier is very high, especially for 3D geometries based VR experiences where high-end end-user devices are needed.

WP3

Content adaptation (T3.3) is a crucial feature to make uploaded media ready to be used in authoring and consuming applications. One special service is the improved 360 transcoding to make 360-degree streaming video available to authoring tools.

WP4

To make media from MediaVerse nodes available to the authoring tools and distribute derivative content through the MediaVerse network, the digital asset management (DAM) must ensure the correct usage of media assets by internal and third-party tools. This includes proper adherence to copy- und usage rights, compensation, and attribution.

WP6

This WP delivers the foundation of the MediaVerse core software components that the authoring tools need to use. Authoring tools developed against a MediaVerse node API must adhere to the specified interfaces.

WP7

As the authoring tools aim at real users, a user centric development and evaluation process is fundamental to build high-level products. Over the course of the project milestones, there will be several opportunities to gather valuable feedback by users to improve further these tools.

Overall, the outline of this deliverable is as follows. In Section 2, we describe the developments for the support of the authoring of VR experiences based on 360 degrees footage. In Section 3, we describe the developments in Fader for 360 VR experiences (T5.3). In Section 4, we provide details on face blurring service for 360 videos, which constitutes a subtask of T5.3. In Section 5, we describe the developments in VRodos for 3D models based VR experiences. In Section 6, we describe the developments for the support of a social media based service for enriching experiences with new content. Conclusions and future work are discussed in Section 7.

2 Support of the Uploading of 3D Models in the MediaVerse Node

Each MediaVerse Node has a Dashboard UI developed under T6.4 that supports the storage and preview of 3D models that can be made available to external tools such as the VRodos. As regards the previewing interfaces, it has a respective section as shown in Figure 2, which allows the users to view the 3D model from all angles. The users can freely rotate it or zoom it by using the mouse or by making gestures with the fingers, in touch screens.

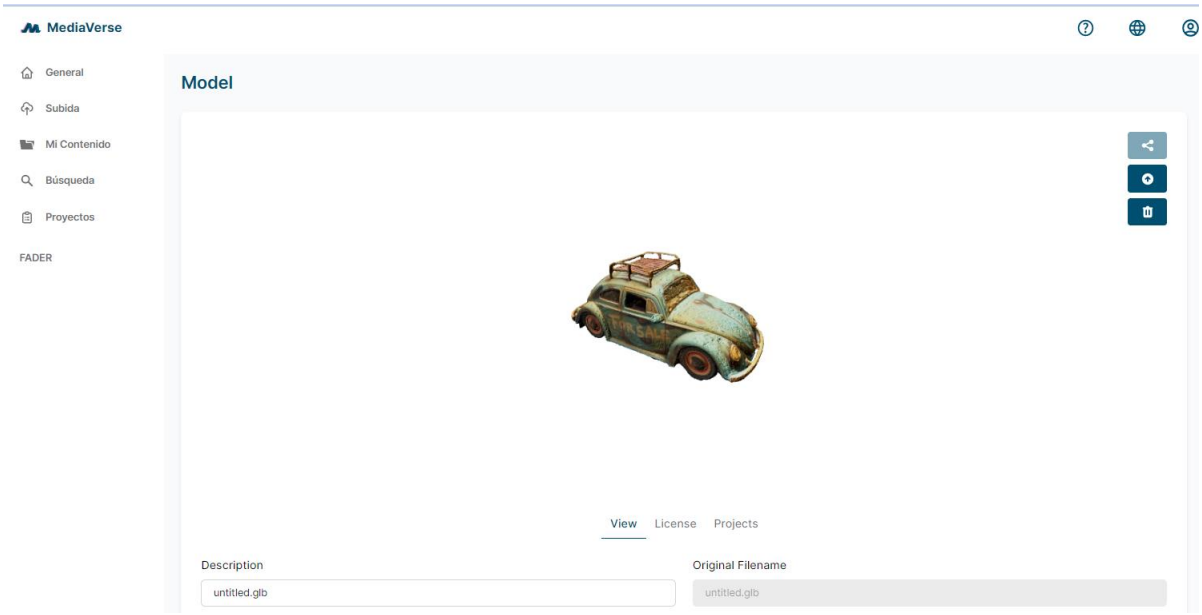

Figure 2: Details of the 3D content section in MediaVerse platform

The 3D previewer has been developed using web technologies, namely:

1. Three.js[2]: A JavaScript library for loading the GLB and OBJ models. It is a lightweight, cross-browser, general purpose 3D library;
2. React Three Fiber[3]: A component that makes Three.js compatible with React JavaScript logic for web pages;
3. React Three Drei[4]: A collection of useful helpers for React Three Fiber.

Although the uploading of 360 videos has been implemented, the user interface for previewing 360 videos is pending. The next important step to be developed for the user interface includes the previewing of 360 degrees videos with functionalities such as:

1. Playing 360 videos in the web browser; and
2. Streaming 360 videos in an adaptive way.

Another development task is the previewing of many 3D models simultaneously, e.g., by making a 180 degrees rotation gif for each model and automatically starting it when the user hovers the mouse onto it.

[2] Three.js, https://threejs.org/

[3] React Three Fiber, https://github.com/pmndrs/react-three-fiber

[4] React Three Drei, https://github.com/pmndrs/drei

3 Fader Immersive Storytelling Toolset

D5.2 - Immersive Storytelling Authoring Tools v1[5] describes the baseline technology of Fader, highlighting its components and functionality. The FADER SaaS (https://app.getfader.com) is developed, maintained, and operated by VRAG. Fader allows content creators a low-level entry into authoring interactive 360-degree stories. Within MediaVerse, the baseline technology is adapted, modernised, and open-sourced to be usable with MediaVerse nodes as an additional media source.

3.1 Application Description

Fader consists of a backend for user management, project management, and media and asset management, processing, and delivery. Fader also consists of a frontend component that is a combined editor and player. While Fader can still work standalone (users can upload media directly), it will allow interfacing with any given MediaVerse node (see section 3.2 Integration with a MediaVerse Node).

Logged in users have access to three additional page sections, Media Library, Projects, and Analytics. Upon login, a user is presented with a project view (see Figure 3), which is initially empty. From there, they can create new projects (known as a story), edit, or remove them. Starting a new story or editing an existing one, takes a user to the editor, the real frontend application of Fader, which is described in more detail in its own section below.

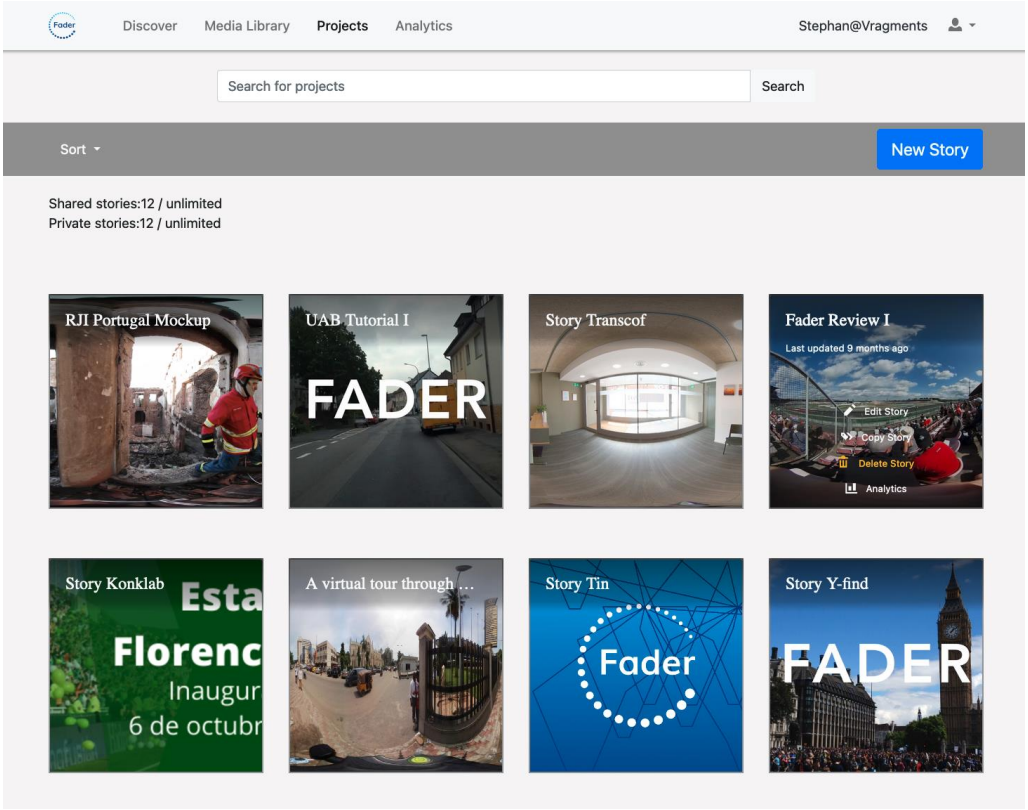

Figure 3: Projects view of a Fader user

[5] https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D5.2_Immersive-Storytelling-AuthoringTools-v1.pdf

The Media Library consists of all media a user has uploaded (see Figure 4). Just like in Discover or Projects, the elements can be searched, filtered, and sorted. Clicking on a tile reveals the asset detail information. The media library allows users to view their assets from a MediaVerse node to be used in Fader projects. An import action on an asset will trigger a processing pipeline that:

- downloads an asset from the MV DAM to create a local asset,
- analyses the local asset (an original local copy of the MV asset) and forwards that metadata to the respective processing tools depending on the asset type (image, video or audio)
- starts the actual transcoding process to transform the original assets into variants that can actually be used and displayed by Fader

Section 3.2.2 of D5.2 describes in detail media processing.

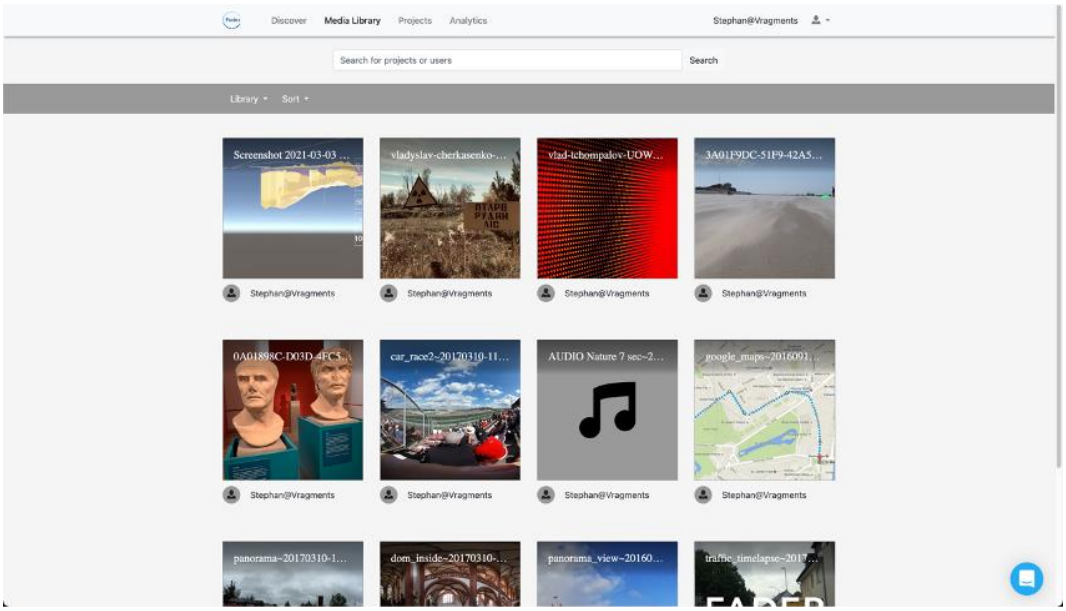

Figure 4: Media Library view of a Fader user

The detailed information of an asset includes a larger preview, some high-level metadata, and the licence information (see Figure 5). A private licence marks an asset as only available for the user. A public licence allows the asset to be used by anyone on the platform. Currently, these are the only licence types, but a more fine-grained licence model can be applied, if the requirements in the project demand such adaptation.

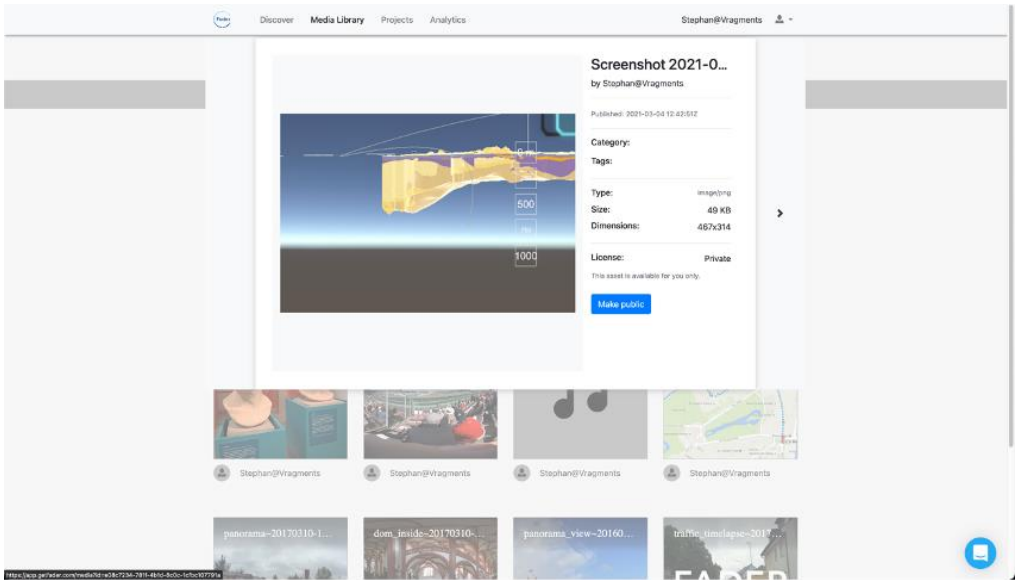

Figure 5: Detailed view of an asset in the Fader Media Library

Creating or editing a new story opens the Fader editor. The Fader editor is the main frontend application that allows users to create interactive 360-degree stories. The top bar contains all story specific settings and options, like setting title and author, adding media (from the library or via upload) and viewing or sharing the story. The bottom bar shows all the story scenes in order, allows users to preview the current scene, give it a name and register it in the scene panel for the player frontend. The right bar contains all media tools for the currently selected scene to add, edit, and remove media assets from the scene.

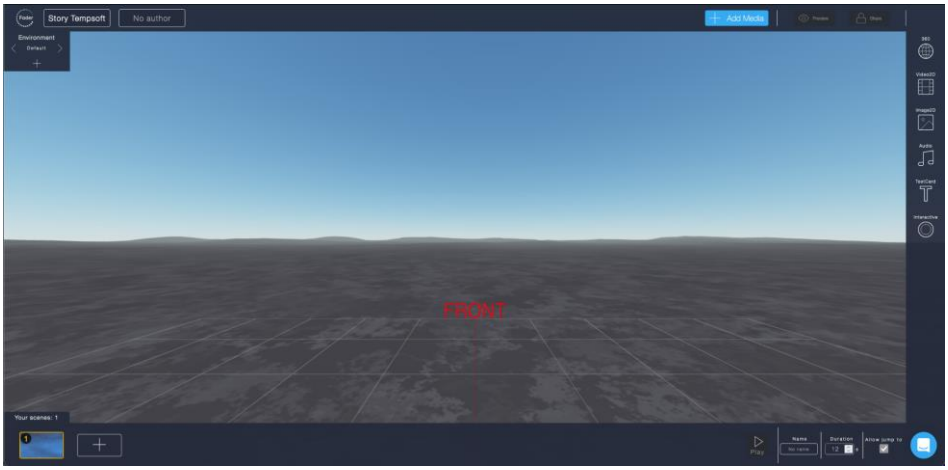

Figure 6: Editor view of a newly created story in Fader

As Figure 6 shows, the first action point, when having a new story open is to import all media a user wants to use in the project. It will not create copies of the assets per story, but will reference them by their asset and asset lease id. The Add Media button opens a modal that shows the media library on the left side, and the used media elements in the story on the right side (see Figure 7). Media can simply be dragged and dropped from the user's own or public library into the story. Users can also upload their media within this screen. A media upload triggers the media processing toolchain described in the backend section. UI feedback is given on the processing steps, success, or failure. Media assets that have been acquired from a MediaVerse node will also appear in this dialog, once their processing has finished.

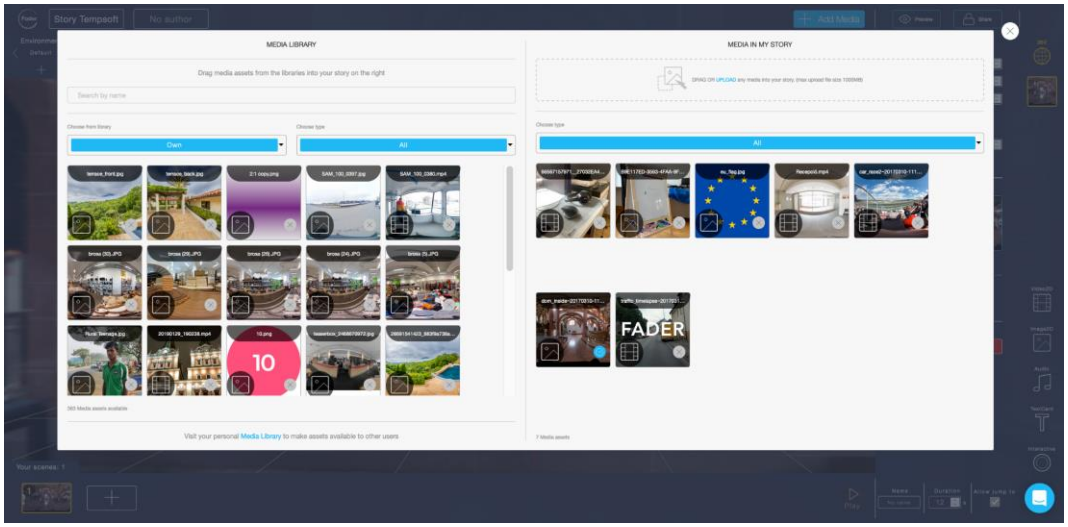

Figure 7: Editor add media view of the Fader editor

Media that is assigned to a story can then be used from the editor toolbar on the right. It denotes components that can be created inside a scene. These components are 360-degree media (images or videos), 2D video, 2D images, audio, text cards and interactive hotspots. Figure 8 shows a 360-degree image used as a scene background. The component has properties that can be edited, such as rotation along all axes and the vertical position. The components content can also be changed by clicking on its preview image to replace it.

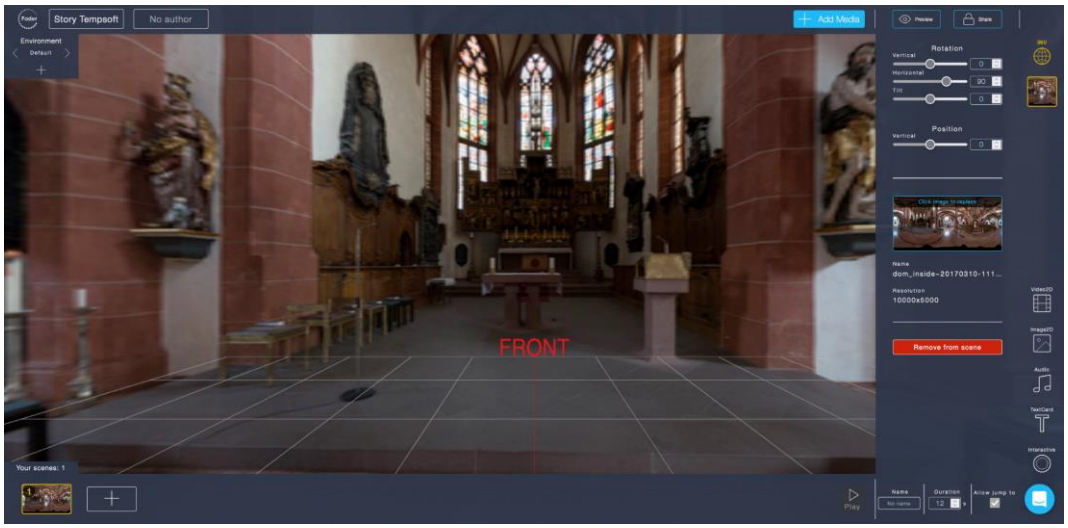

Figure 8: Editor view of a scene in Fader consisting of a 360-degree background from an image asset

Fader contains a predefined and extensible set of 3D backgrounds. New templates that allow for predefined content placement are being developed with partner DW to satisfy requirements for journalistic content. Figure 9 shows a 2D video being placed inside a 3D scene that is generated procedurally, resembling a rocky environment.

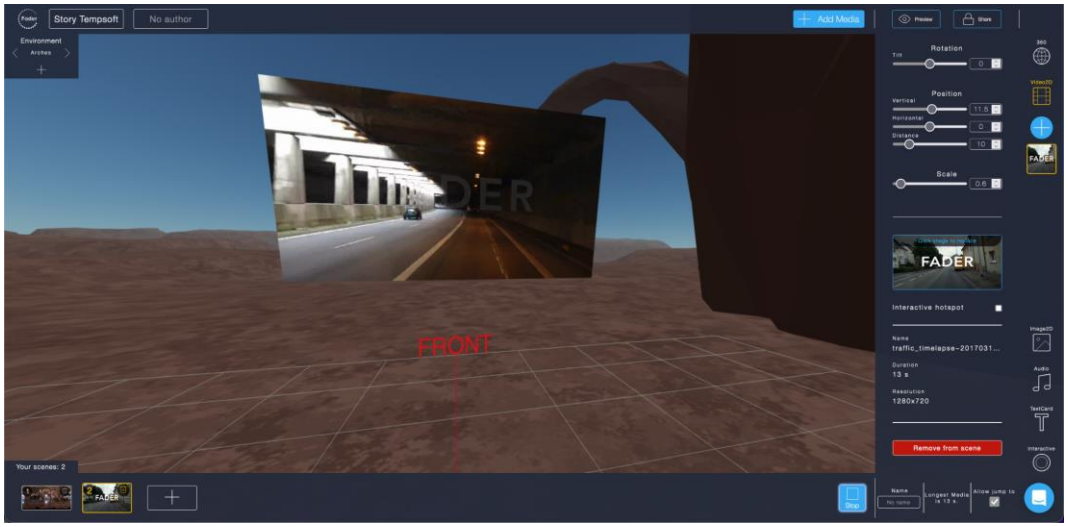

Figure 9: Editor view of a scene in Fader consisting of a 2D video overlaid on a 3D background

2D elements, such as videos or images can be also displayed as interactive hotpots to open only on user input. To do so, users simply mark the asset as an interactive hotspot. These 2D hotspot elements can also have captions.

Figure 10: Editor view of a scene in Fader consisting of a 2D video interactive hotspot in open mode with captions shown in world space

Interactive 2D elements have two view modes based on how a story is viewed - either in VR mode with a headset or in magic window mode on a desktop or mobile device not running in VR mode. In VR mode, 2D elements are displayed in the scene itself (world space) - just as shown in the editor (Figure 10). In magic window mode, 2D elements are displayed in the screen space to improve visibility of content and captions (Figure 11).

Figure 11: In magic window mode, interactive 2D images or videos are centred flat in screen space for optimal viewing

To make stories interactive and feature rich to be used for various purposes from linear stories to even quizzes, one can use interactive hotspots that function as links between scenes. As with any other component, interactive hotspots can be freely positioned and scaled (Figure 12). Their appearance can be configured with any image available. The scene they should link to is defined in by providing a scene from all existing ones except for the scene that contains the hotspot (no self-reference).

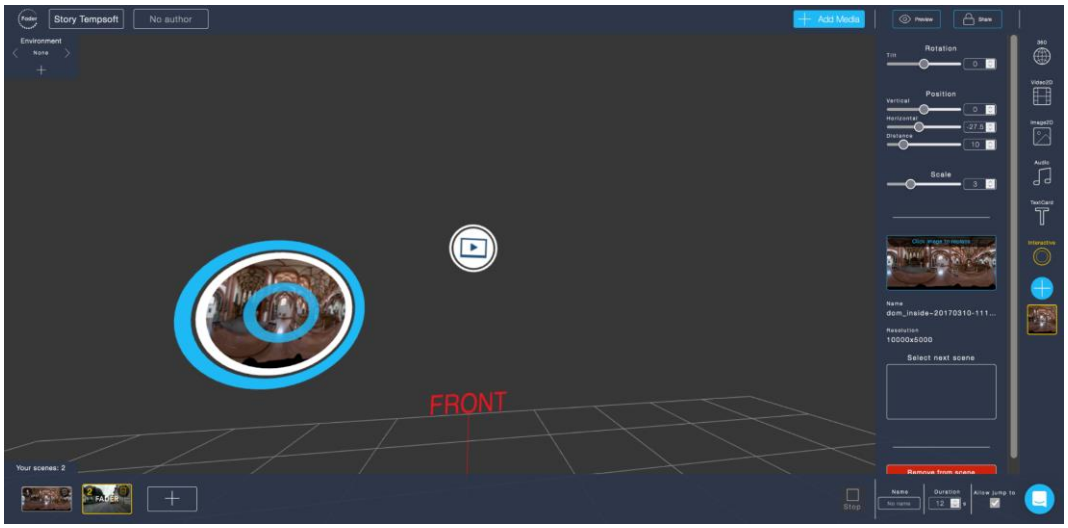

Figure 12: Editor view of a scene in Fader consisting of a 2D video interactive hotspot in open mode with captions

The hotspots work in desktop or magic windows mode as well as in VR mode with help of a click timer.

Once a user is happy with the story, he or she can share or publish it in the Discover section of Fader. Users can select both options via radio buttons to set it either completely private, link share (to anyone that has the link) or public to appear on Fader's public list of projects (Figure 13).

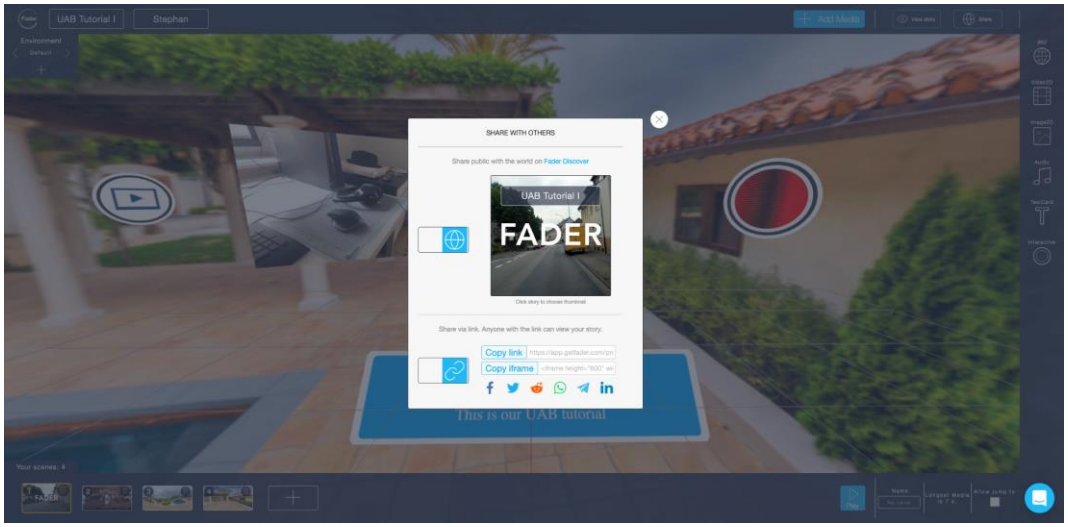

Figure 13: Editor view of story sharing options

A shared story has its distinct URL that is routed through the Fader player. It preloads all necessary assets via the project manifest file (e.g., it determines the story preview image, all scenes and components plus their individual assets).

Figure 14: Loading screen of a Fader story

The loading screen contains the play button in the centre, once the scene is fully loaded (Figure 14). Before, a loading status with estimated percentage of assets loaded is shown. To the lower right, there are options to share the story link with several social networks. Based on the target chosen, specific metadata is created to build a meaningful post with image preview, title and description.

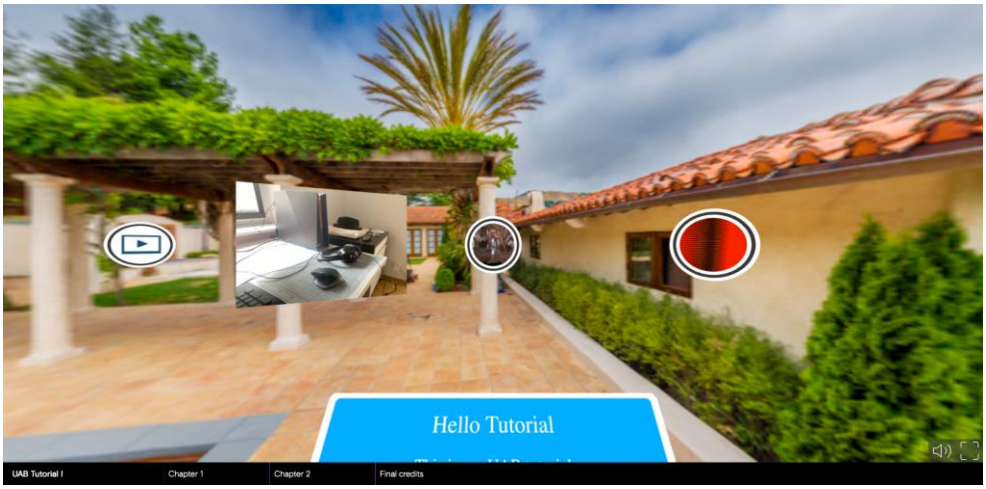

Figure 15: View of a Fader story playing in desktop mode showing interactive hotspots, a running video and a text card

Upon clicking the play button, the viewer enters the first scene in the story. Based on the scene content, it either plays for a default time, its maximum media length or runs until the user has selected one of the interactive hotspots to change scenes (Figure 15). Authors can choose which scenes should be directly reachable. If this feature is selected for a scene, they are shown in a navigation bar at the bottom (Figure 16). If no scene is selected to be shown there (for example for a quiz), the navigation bar is completely hidden.

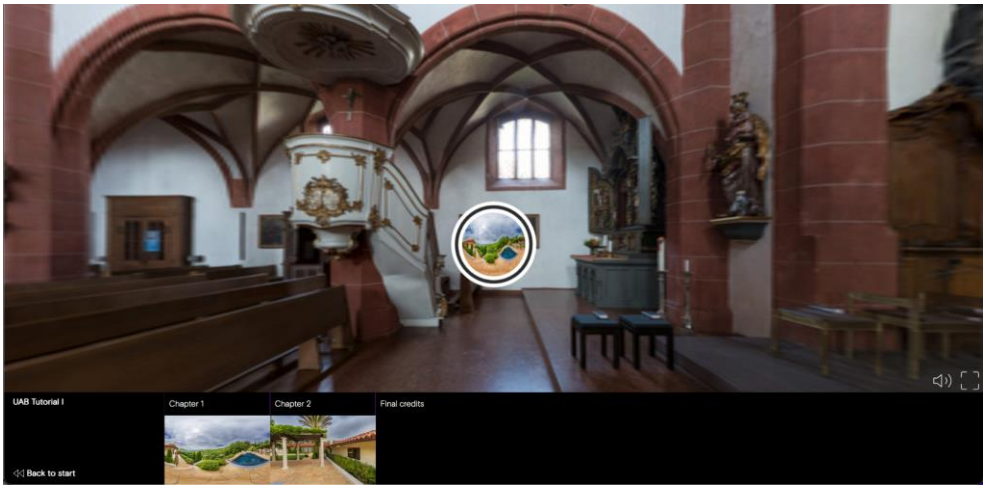

Figure 16: Player view of a Fader story with the scene navigation bar open.

3.2 Integration with the MediaVerse Node

Fader can be used in different scenarios that support variable use cases and deployment requirements. As it is available in a configurable Docker container, it can be deployed on virtually any hardware that meets the minimum requirements for data processing and a specified number of concurrent users. By giving administrators maximum flexibility, it can be deployed as:

- an internal service inside an institution,
- an internal service for production and an external service for publication of contents,
- with or without using a MediaVerse node (a default node can be configured, or users can connect to an MV node of their preference).

Fader offers multiple authentication mechanisms to maximise its value as an open source tool. As Fader has its own user account system, a local account is required. Users can choose to sign in to an MV node that they are registered to and a local account will be created for them with access token management. The access token is used to retrieve MV projects and assets that the user in the MV node has access to (Figure 17).

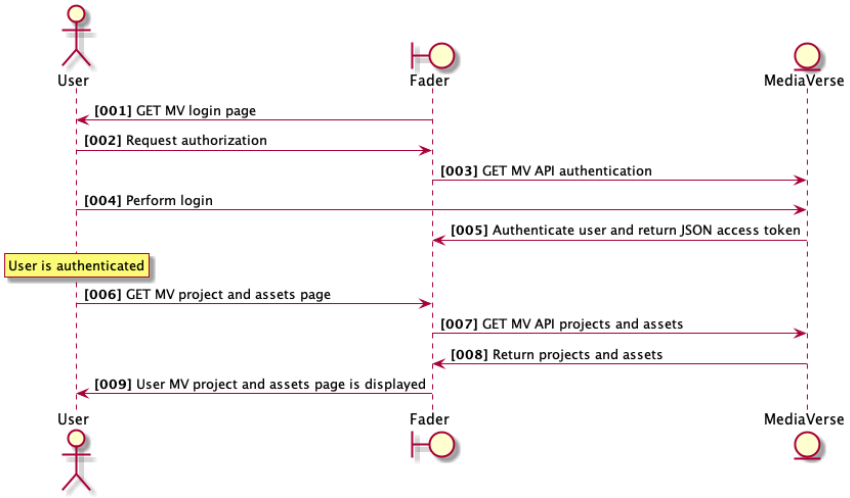

Figure 17: Sequence diagram of user login and retrieval of assets from MediaVerse

A Fader project is a collection of asset references with metadata to feed into a 3D scene. The 3D content is rendered using Three.js and Aframe. While Three.js is a library to render 3D content in WebGL, Aframe abstracts this into a component-entity-model for simple HTML tags.

Alternatively, other authentication providers can be used via OAuth, for example using GitHub or Google OIDC (see Figure 18). In that case, users do not have access to an MV node, unless they connect their account to it.

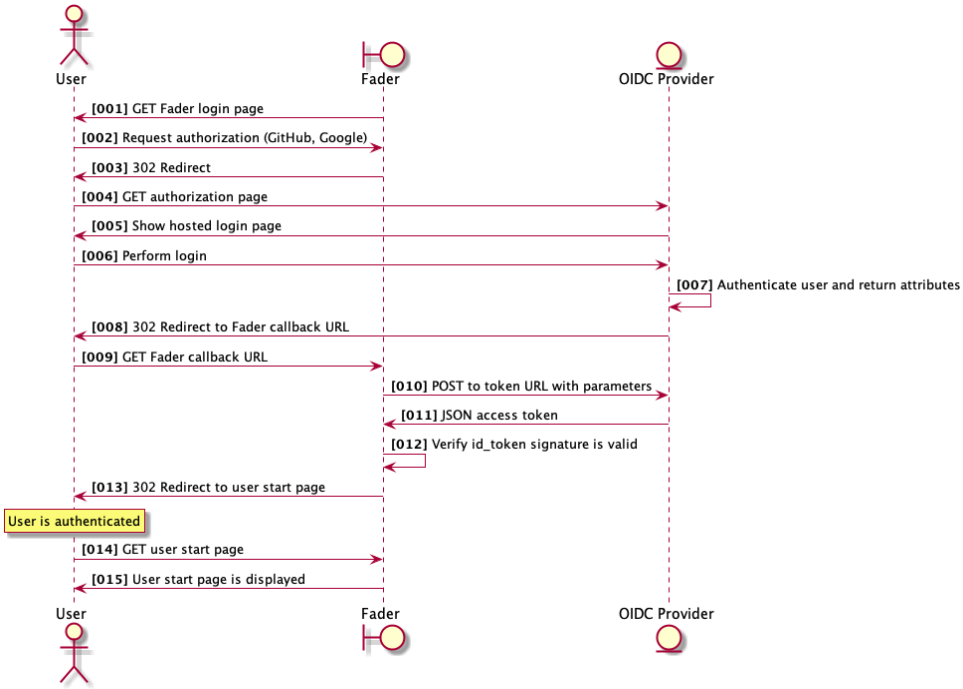

Figure 18: Sequence diagram of user login to Fader via an OIDC provider

One still needs to define how a Fader story can be shared with the MediaVerse network. Since a Fader story consists of a manifest file, a player frontend and the transformed media assets, it needs an endpoint that contains all necessary code and assets to run a Fader story. By default, this is the Fader endpoint itself, which is outside of a MediaVerse node container. A proposed solution is to define a special asset type in the MediaVerse DAM that contains:

- A URL to the story in the form of https://example.com/projects/b61becec-eeb7-4f2f-936d-25b8cabe4df1/publish
- An identifier for the 3rd party authoring tool
- A freely chosen name of the project (can be the same as the story name in Fader)
- The owner or author as identified per the MediaVerse node
- A list of used assets from MediaVerse with their references

This could also be achieved through simply uploading a text file in JSON-format that contains the references, but one would lose the ability to directly track asset usage within MediaVerse.

3.3 Addressing MediaVerse Requirements for Fader

This is an update on Table 3.1 - Developments to support the requirements across the technical components from D5.2.

UX-02 / Content Information will be easily visible.

This requirement refers to the display of asset information through Fader and will help users pick the right assets for their immersive story. Since the user is authenticated against an MV node, all available asset information can be acquired from the MV node DAM and presented to the user upon asset list and asset detail views.

COLL-02 / The dashboard provides a concise overview of various types of information.

From the Fader authoring tool, only limited information needs to be available. Main API endpoints for Fader are the user, project and asset endpoints to acquire information and data and write information back on the project status, e.g. used assets to make that available to usage tracking within MediaVerse.

COLL-04 / Project participants communicate via simple chat.

This feature is no longer directly associated with the authoring tools and should rather be a support function of a MediaVerse node itself.

COLL-05 / Project participants can add notes in the project.

This feature is no longer directly associated with the authoring tools and should rather be a support function of a MediaVerse node itself.

PLAY-01 / The player enables display/overlay of subtitles from .srt files.

Uploaded video to a MediaVerse node can make use of the accessibility features provided by STXT Racu application. Using this, subtitle text files (.srt) are returned to MediaVerse and can be referenced through the original video asset. When such an .srt file exists for a given video, it can be used by Fader to display the text inside a 360 story as an overlaid text card. The import of such an .srt is done through the Media Library described in 2.2.1

PLAY-02 / Accessibility extras can be switched on and off.

The display of subtitles in a story should be an option in the player - similar to a "CC" button on most web video players. This feature is still in development and will trigger a certain class of scene components to render or not.

BRKR-04 / Incorporating segments will have implications on the usage rights of the project.

A user is generally only able to use assets (from their MediaVerse account) in a Fader project, that they have obtained the rights to download them. Since these assets need transformations for technical reasons and they will be part of a collage, only such assets should be eligible to download and use in Fader that have the correct rights to use them in that fashion.

AUTH-01 / Mobile and desktop interfaces.

The Fader backend, as well as the editor and player are fully working on desktop interfaces. With mobile devices and VR, certain restrictions apply, such as default muting of audio content and necessary device capabilities to display VR content with rotation sensors active.

AUTH-02 / MediaVerse enables content creators to do basic editing on audio, video, 360 Video.

Basic editing of assets should be supported outside of Fader. Fader operates on finalised media assets.

AUTH-03 / External authoring tools can be connected with MediaVerse for easy updates.

A Fader user can log in to their MediaVerse account. Having done this, they can acquire updates on assets within their projects and update their stored information on a Fader project accordingly.

AUTH-04 / The authoring UIs need to support various languages.

Fader is currently available in English, French and German. The localization can be extended, provided there is capacity to translate within the consortium.

AUTH-05 / Creators can add and edit various accessibility enhancements to content.

Fader is designed to support subtitles on videos, using generated .srt-files from a MediaVerse node.

XR-02 / A 3D Scene can be exported as a video clip.

Due to the technical complexity of this issue (performance of capture and rendering) and various logical fallacies with interactive, non-linear content, a video export is not feasible. If users want to capture their experience, they should rather use screen capture software, which will yield a better result.

XR-05 / 360-video can be enhanced by adding objects/assets.

Fader supports adding various interactive and non-interactive content on top of 360-degree video. This is described in section 2.2.1.

XR-06 / 360-video can be enhanced by hotspots with different functions.

Fader supports adding various interactive hotspots on top of 360-degree video. This is described in section 2.2.1.

RGHT-03 / The system must be able to trace content sources (inside MV).

Fader relies on the MV-DAM to trace content sources and displays this accordingly to the user. A suggestion to track asset usage for a Fader project is given in section 2.2.2.

MNTR-01 / Monitor the consumption and re-use of content.

Since content used in a Fader story is downloaded and transcoded, there is a disconnect from the original asset to its derivative works. While statistics of a Fader story are generated and stored, this offers only relative monitoring on a project level, not on individual asset usage. However, if we look at the analogy of video production using assets from an MV node, this is quite comparable. Assets are downloaded and used in a new video. At that point, a direct link to the assets and their usage in a new video is lost. When the result is uploaded to MediaVerse (a Fader JSON project file or a video file in the latter scenario), asset references must be tagged inside MediaVerse. A suggestion to track asset usage for a Fader project is given in section 2.2.2.

4 Face Blurring in 360 Footage

Hundreds of millions of online social network users present themselves, communicate, and share thoughts and pictures every day. In 2016, Facebook users alone uploaded more than 250 billion photos, an amount that keeps increasing with ever-growing rates. More recent evidence[6] suggests that 3.2 billion images and 720,000 hours of video are shared online daily. The data shared on social networks and other digital platforms may include sensitive details, which generates privacy issues such as unintentional facial recognition, inference attacks, identity theft or profiling risk.

One approach to control photo/video content disclosure is through face blurring. Face blurring refers to the procedure that is designed to protect the privacy of individuals by algorithmically blurring their faces wherever they appear in an image or video. The technique is used widely across industries from street mapping and directions, to news reporting and social media.

In order to realise face blurring for images and videos there are two steps that should be taken into account. The first one, which is the most challenging, concerns face detection. The vast majority of existing face detection algorithms are based on Machine Learning (ML) in order to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands.

The detection procedure initially leads to the identification of the smallest bounding boxes that enclose the detected faces. Then, a second step takes place that is referred to as face alignment. Face alignment is a computer vision approach for identifying the geometric structure of a human face in digital images. Given the bounding box (location and size) of a face, it automatically determines the shape of the face components such as eyes and nose. The typical procedure followed in face alignment is based on the iterative adjustment of a deformable model, which encodes the prior knowledge of face shape or appearance and takes into account the low-level image evidence. Ultimately, face alignment leads to a fine-grained localization of the face that is present in the images/videos. Finally, a blurred eclipse can be applied on the identified mask to blur the facial characteristics.

Face blurring constitutes an important feature of MediaVerse's platform since a particular case study (i.e., scenario 1.2 - Immersive Journalism; as it is described in D2.1 - Use Cases and User Requirements[7]) deals with journalists publishing 360 degrees footage content. This feature will potentially work in regions where witnesses are punished simply for talking to the media, or when covering events that anonymization is of paramount importance. Hence, face blurring paves the way for documenting the events in the manner they occur, while the vulnerable news sources remain protected. It should be noted that face blurring will be detailed as a particular requirement in D2.3 which will document the updated Use Cases and User Requirements.

4.1 State of the Art

Face detection is a long-standing problem in computer vision with many applications. Following the pioneering work of Viola and Jones (2004), which was probably the first notable effort in face detection, numerous face detection algorithms have been designed. Namely, the cascade face detector that utilised Haar-Like features and AdaBoost to train cascaded classifiers. Prior to the deep learning era, this approach appeared as a golden

[6] https://theconversation.com/3-2-billion-images-and-720-000-hours-of-video-are-shared-online-daily-can-you-sort-real-from-fake-148630

[7] https://mediaverse-project.eu/wp-content/uploads/2021/04/D2.1-V1.0.pdf

standard since it achieved good performance with real-time efficiency. However, several works [Yang et al., 2014; Pham et al., 2010; Zhu et al., 2006] pointed out that these detectors degraded significantly in real-world application with larger visual variations of human faces even with more advanced features and classifiers. Later, some Convolutional Neural Network (CNN)-based face detection approaches were proposed [Krizhevsky et al., 2012; Sun et al., 2014; Yang et al., 2015], however due to the complex structure of CNNs these approaches are time costly in practice and lack inherent correlation between facial landmarks localization and bounding box regression.

In an effort to overcome such issues, the single-shot anchor-based approaches [Najibi et al., 2017; Zhang et al., 2017a; Tang et al., 2018; Li et al., 2019; Ming et al., 2019; Deng et al., 2020; Liu et al., 2019; Zhu et al., 2020] have been recently introduced and managed to achieve the most promising performance. In particular, on the most challenging face detection dataset, WIDER FACE (Yang et al., 2016), the average precision (AP) on its Hard test set has been boosted to 92.4% by TinaFace (Zhu et al., 2020). Even though TinaFace (Zhu et al., 2020) achieves impressive results on unconstrained face detection there are two downsides. First, the employment of large-scale testing has an enormously big computational cost for real-time applications. Moreover, its generic object detection backbone, head and neck design, make it sub-optimal and redundant for face detection applications. Since directly taking the backbone of the classification network for object detection is sub-optimal, some works [Liu & Tang, 2020; Liu et al., 2019] have tried to reallocate the computation across different resolutions. However, most of them still only focus partially on the optimization of only one of the backbone, head or neck.

To achieve a reliable face blurring procedure on 360 videos, we took advantage of the model proposed by Guo et al. (2021), namely Sample and Computation Redistribution for Efficient Face Detection (SCRFD), which uses sample and computation redistribution for efficient face detection. Guo et al. propose two improvements on the design of face detection, compared to TinaFace, which constituted the most reliable solution at that time. These efficiency improvements are conditioned on: a) the testing scale bounded at the VGA resolution (i.e., 640), and b) there is no anchor tiled on the feature map of stride 4. Particularly, the redistribution of positive training samples across different scales of feature maps is investigated in order to deal with the smaller testing scale that is used. Then, the computation redistribution across different components (i.e., backbone, neck, and head) is explored, given a predefined computation budget in order to figure out the relationship between computation distribution and performance from populations of models.

By examining SCRFD in more detail, during training data augmentation, square patches were cropped from the original images with a random size from the set [0.3, 1.0] of the short edge of the original images. To generate more positive samples, the random size range was enlarged from [0.3, 1.0] to [0.3, 2.0]. Even though this process introduced more extremely tiny faces (e.g., $< 4 \times 4$) under the large cropping strategy, these ground-truth faces were neglected during training due to unsuccessful anchor matching. With more training samples redistributed to the small scale, the branch to detect tiny faces could be trained more adequately.

Furthermore, in order to design a better network for efficient face detection the computation search space of Neural Architecture Search (NAS) should be reduced. NAS is a technique for automating the design of artificial neural networks in order to outperform the efficiency of hard-designed architectures. In this direction, some simplifications regarding the hyperparameters of the components of the network were introduced which made the search space more straightforward. Afterwards, 320 models got selected from the space with random sampling in the target complexity regime and through these models' statistics (computation ratio of a particular component, corresponding performance) the likely range in which the best models fall got estimated following the empirical bootstrap (Efron & Tibshirani, 1994).

To further decrease the complexity of search space, the network structure search was divided in the following two steps:

- SCRFD1: Search the computational distribution for the backbone only, while fixing the settings of neck and head to the default configuration.
- SCRFD2: Search the computational distribution over the whole face detector (i.e. backbone, neck and head), with the computational distribution within the backbone following the optimised SCRFD1.

The computation ratio between the shallow (i.e., stem, C2, C3) and deep stages (i.e., C4, C5) of the backbone is represented in Figure 19. Based on the observation from these search results, it is obvious that approximately 80% of the computation is reallocated to the shallow stages.

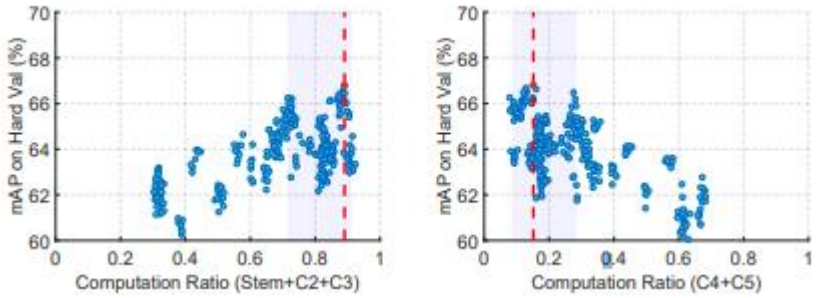

Figure 19: Computation redistribution between the shallow and deep stages of the backbone under the constraint of 2.5 Gflops. (a) Stem+C2+C3 ~ (72%, 91%); (b) C4+C5 ~ (9%, 28%)

The next step consisted of searching for the best computation distribution over the backbone, neck and head following the distribution results from SCRFD1. This step introduces three degrees of freedom (i.e., output channel number n for neck, output channel number h for head, and the number of 3×3 convolutional layers m in head). Random sampling in this new search space is repeated until 320 qualifying models in the target complexity regime (i.e., 2.5Gflops) are obtained. Figure 20 shows that most of the computation is allocated in the backbone, with the head following and the neck having the lowest computation ratio.

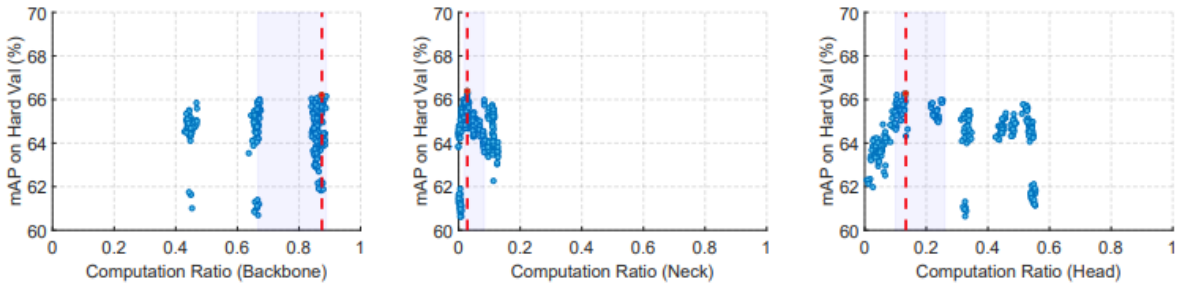

Figure 20: Computation redistribution and architecture sketches under the constraint of 2.5 Gflops: (a) Backbone ~ (67%, 88%); (b) Neck ~ (1%, 7%); (c) Head ~ (10%, 26%)

By employing the proposed two-step computation redistribution method, a large amount of capacity allocated to the shallow stages is revealed, resulting in an AP improvement from 74.47% to 77.87% on the hard validation set of WIDER FACE (Yang et al., 2016).

To demonstrate its performance, the SCRFD approach has been tested on the WIDER FACE benchmark, which splits its images in three categories according to their difficulty of identifying the faces. As can be seen in Figure

21(a-c), SCRFD's results are outperforming other state of the art methods [Liu et al., 2020; Zhang et al., 2017b] in all three categories (i.e., variations of the validation set), not only in accuracy, but also in computation efficiency. More specifically, SCRFD-34GF surpasses TinaFace by 3.86% while being more than 3× faster on GPUs. In addition, the computation cost of SCRFD-34GF is only around 20% of TinaFace. As SCRFD-34GF scales the depth in the earlier layers, it also introduces fewer parameters, resulting in a much smaller model size (9.80Mb).

For real-world face detection systems, high precision (e.g., > 98%) is required to avoid frequent false alarms. As shown in Figure 21(d), SCRFD-2.5GF obtains an AP (53.7%) that can be directly compared to TinaFace (53.9%) when the threshold score is set to achieve prevision higher than 98%, while the computation cost is only 1.46% and the inference time is only 10.8%.

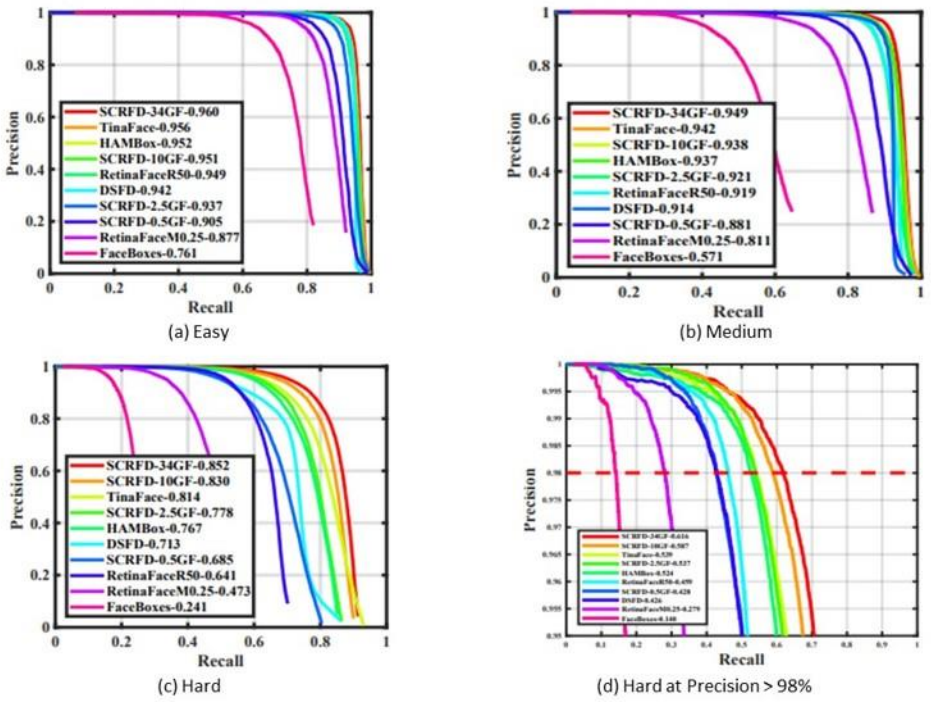

Figure 21: Precision-recall curves on the validation set of WIDER FACE dataset.

4.2 Implementation and Qualitative Results

We provide qualitative results based on videos provided by VRAG and DW. Initially, we separated the videos into sequences of frames, which were given as input in the SCRFD model. The model's output for each frame is a bounding box formatted as [x, y, width, height]. Finally, these features are used to plot blurred eclipses over the faces that are detected in each frame and the frames are placed back in a sequence to reconstruct the blurred 360° video. We note that, although frame-by-frame blurring is less time efficient than video-based approaches (e.g., Nagendra et al., 2015) it outperforms the latter in terms of accuracy. Hence, it was our strategic choice to employ a less efficient, yet more robust, approach in order to support this privacy sensitive task.

In order to exploit the SCRFD architecture, the 360° frames were transformed using the equirectangular projection. Although it is expected that projection will affect the model's performance in 360 videos where face distortion is the direct aftereffect of the projection, Sitzmann et al. (2018) showed that salient objects (such as faces) are mainly concentrated in the equator of 360-content. The important advantage of the equator in 360-

content is that it is the area of the image/video with the minimum distortion, which could be considered as a good indicator to apply mainstream-media face blurring techniques on 360-media.

Finally, our implementation was evaluated qualitatively on some sample videos that were provided by VRAG. By examining these videos, it becomes evident that the employed approach is capable of detecting the vast majority of the faces that are included in the videos while working efficiently in low lighting situations. The most evident occasions of undetected faces, is when they are extremely distant from the camera and the corresponding area occupied is of a few pixels (i.e., low scale). However, it is important to mention that in these cases the face characteristics can be hardly distinguished. Therefore, privacy concerns are weakened and the necessity to perform face blurring is alleviated. Below we provide some examples for the face blurring mechanism that is employed for the MediaVerse purposes (Figures 22-25).

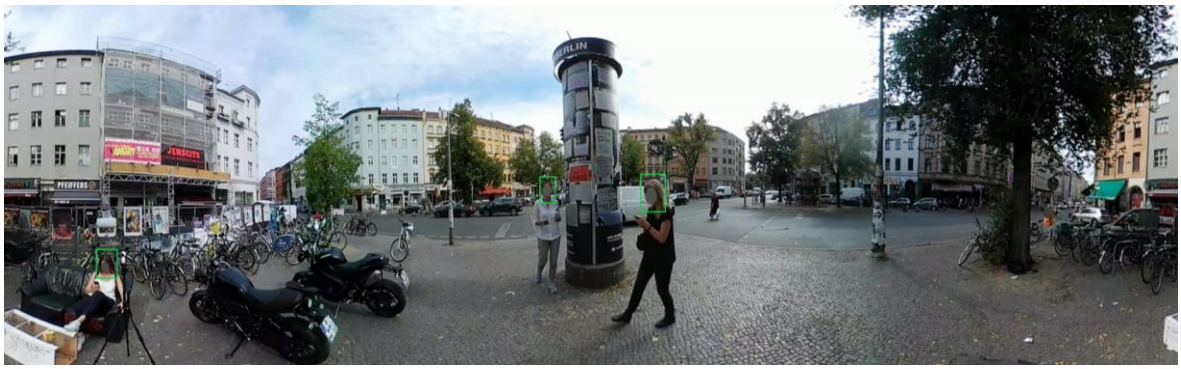

Figure 22: Equirectangular projection of a frame in normal lighting conditions. All of the faces have been blurred.

Figure 23: Equirectangular projection of a frame in normal lighting conditions which contains multiple faces in multiple scales. Every face with identifiable characteristics has been blurred. A few low-scale faces are being missed (see below)

Figure 24: Zoomed and cropped version of the previous equirectangular projection. Out of the four low-scale faces, one (non-identifiable characteristics) is missed.

Figure 25: Equirectangular projection of a frame in low lighting conditions. Result: The vast majority of the faces are blurred (especially those at high scales)

The process of face blurring boils down to the identification of the presented face(s) in a given image, registered as bounding box(es) followed by a blurred eclipse applied on the identified bounding box(es). Here, we employed the SCRFD approach for the face detection process, which has proved to be efficient in both terms of running time and robustness with respect to the identified faces. Additionally, we provide qualitative results based on in-house videos created by VRAG/DW that validated the efficiency of SCRFD in this type of video format. Another aspect that we did not include in this deliverable is the potential distortion in faces due to the equirectangular projection in 360^{o} videos. To resolve this issue, we aim to distort existing datasets using transformations that resemble the equirectangular projection of 360^{o} images. Then, the distorted images will be employed in order to fine-tune existing neural network architectures, and more specifically the SCRFD. This will increase the detection accuracy for faces that are potentially located away from the equator, hence being highly distorted.

4.3 Usage in Authoring Tools (Fader)

Creation of a face blurred version of a 360-degree video is done on the MediaVerse platform itself. The process will create a new asset different from the original asset, but with the association to it being kept. This asset, if added to a project, can then be acquired by Fader through downloading and processing as described in Section 3. Users can freely choose which version of a video they would like to use. Keeping this within MV itself allows for greater flexibility with future authoring tools.

5 VRodos for Authoring VR Experiences Based on 3D Geometries

In the previous related deliverable, D5.2 - Immersive Storytelling Authoring Tools v1[8], we presented VRodos, a web platform that allows users to create multi-player WebVR experiences through a web based 3D authoring tool as shown in Figure 26. The web libraries used are the state of the art libraries, namely Three.js[9], Aframe[10], and Networked-Aframe (NAF)[11] that allow creating VR experiences for any kind of device that has a web browser, namely smartphones, tablets, PCs, and Head Mounted Displays.

Figure 26: Year 1 developments on VRodos: a) authoring tool; b) created user experience targeting high-end PCs

However, VR technologies have a great demand on processing power, i.e., require high-end versions for all the aforementioned types of devices. As we have discovered, after a discussion with a focus group (FG) of stakeholders in music and theatre of the broad area of Thermi-Greece as provided in WP7, most of the end-users that want to participate prefer to use their smartphones, as the majority of them does not have a high-end PC

[8] https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D5.2_Immersive-Storytelling-AuthoringTools-v1.pdf

[9] Three.js, https://threejs.org/

[10] Aframe, https://aframe.io/

[11] Networked-Aframe, https://www.npmjs.com/package/networked-aframe

or headset for 3D rendering. Moreover, an additional comment is that they do not want to exhaust the processor and the battery of the smartphone with such a high demanding application as in most of the times, the smartphone becomes hot after 10 minutes of 3D rendering. Emphasis has been also given to the high quality of graphics in the final result as FG participants consider this feature as of major importance. In order to cope with this contradiction, we have devised a custom web-based streaming kernel that performs rendering at a high-end PC in one of the peers participating in the production (called as the Director), and streams back the rendered scene stream to the peers' smartphones that we refer to as Actors. The final result is a virtual production movie or a live movie stream. On this basis, it can be either recorded in the Director's PC and in order to be uploaded to MediaVerse node, or it can be streamed live to social media such as Facebook or YouTube.

In total, four types of users are involved, namely:

1. the System Administrator who is responsible for the set up and maintenance of the system;
2. the Graphics Designer who is responsible for uploading the 3D models and setting up the 3D scenes;
3. the Director who is responsible for the coordination of the actors as well as for the rendering of the scene in a high-end PC; and
4. the Actor who participates through a smartphone.

In Section 5.1 we describe the application. In Section 5.2, technical details about the multi-playing technology and our contributions are described. In Section 5.3, integration activities with the MediaVerse Node are provided. In Section 5.4, the initial requirements coverage is provided. Finally, in Section 5.5, we provide a qualitative comparison with similar tools.

5.1 Description of the Application

We present the overall architecture of the application in Figure 27. The proposed system is divided into four logical phases, namely Setup, Authoring, Playing and Exploitation Phases.

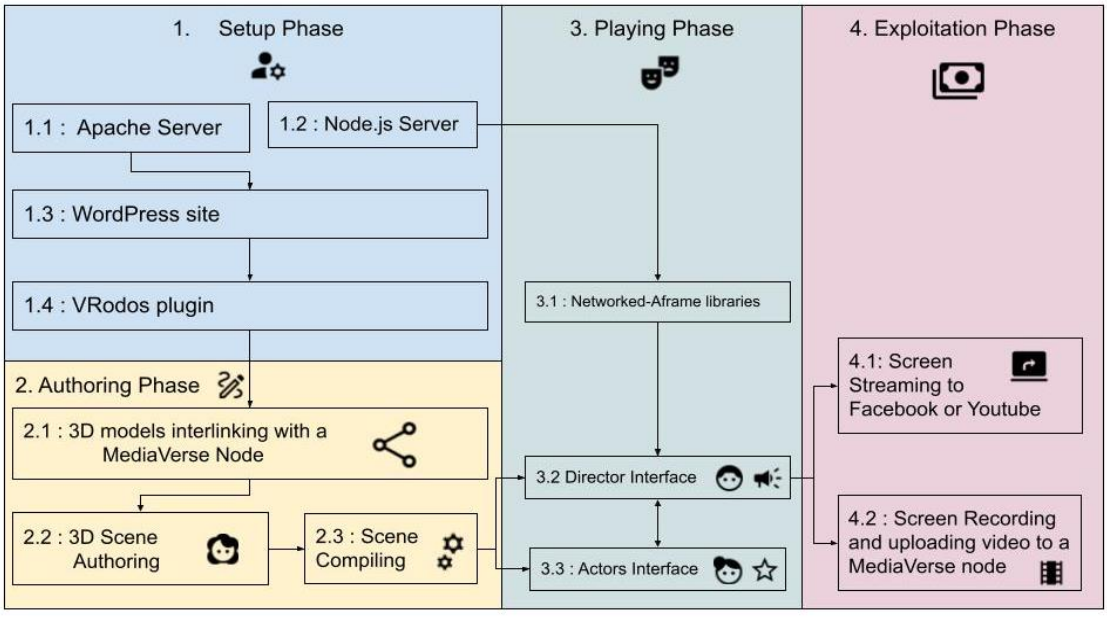

Figure 27: The system is divided in four logical phases

5.1.1 Setup Phase

In Phase 1, the system is being set up. Towards this goal, the following steps should be performed:

1. **System:** A server (Linux, e.g., Ubuntu 20.04 LTS[12]) must be set up. A Windows 10 system can be also used.
2. **Server 1:** An Apache Server[13] technology for serving PHP web pages should be installed.
3. **Database:** A MariaDB[14] database for storing data should be installed.
4. **WordPress:** Next a website based on WordPress software has to be setup[15].
5. **Server 2:** Also a Node.js server should be installed to Ubuntu or Windows which is necessary for the real-time transfer of data in multi-playing experiences.
6. **VRodos plugin:** Then the VRodos plugin has to be installed to the WordPress site. This plugin is the main output of T5.5 and it can be found in the following Github repository[16] as open source software under Apache 2.0 license.

VRodos plugin contains the back-end functionalities, the front-end interfaces, and any software libraries needed such as Three.js and Networked-Aframe. Installation instructions are provided in the GitHub link. A site instance is created at CERTH premises, and it is publicly accessible using this link[17].

5.1.2 Authoring Phase

The Director or the Graphics Designer of the platform should log in to VRodos. They can use the same credentials as in a MediaVerse node in order to gain access to their uploaded models in that MediaVerse node. Then he or she can set up a Project in VRodos where his or her 3D scenes can be designed. An example is provided in Figure 28. On the left, a list of existing projects can be found.

- The ASSETS button shows a list of assets for the selected project;
- The 0 humans icon can assign privileges to other users to make edits in the system;
- The 3D_EDITOR button goes to the 3D scene editing interface;
- The deletes the project.

On the right side panel, a new project can be created by providing only a short title.

[12] Ubuntu 20.04 LTS: https://releases.ubuntu.com/20.04
[13] Apache Server: https://www.apache.org/
[14] MariaDB: https://mariadb.org/
[15] WordPress: https://wordpress.com/
[16] VRodos code at Github : https://github.com/VRodos/VRodos
[17] VRodos demonstration instance: https://vrodos.iti.gr

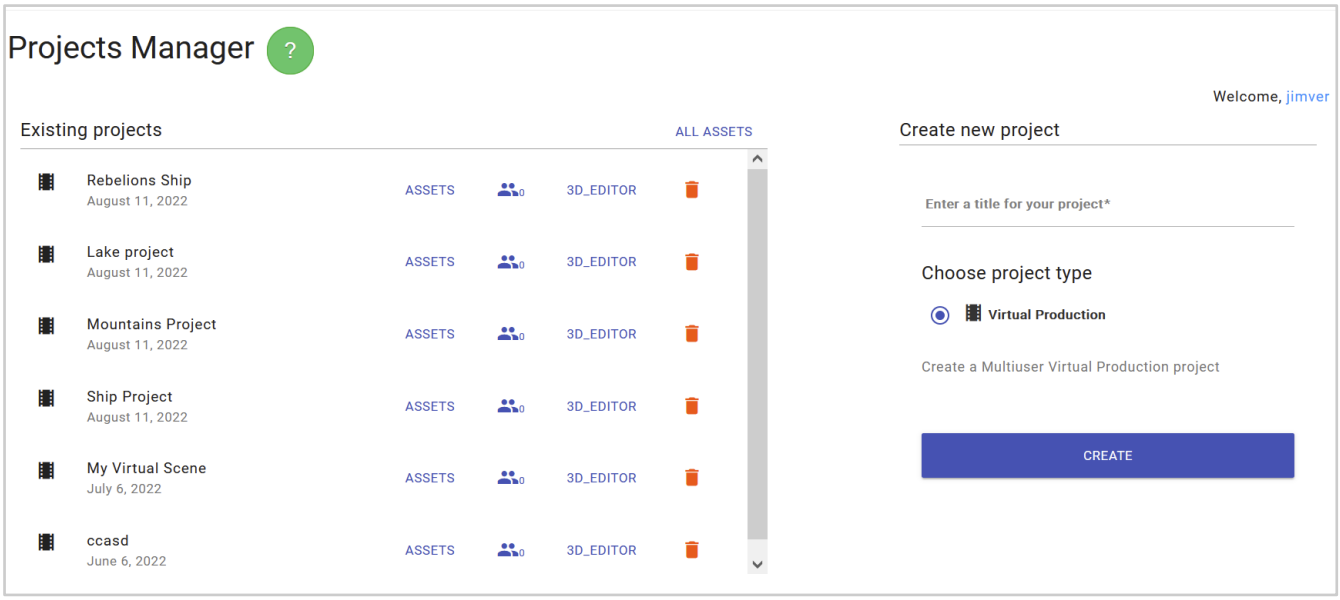

Figure 28: The project manager allows the grouping of scenes

The 3D_EDITOR leads to the scene editor as shown in Figure 29. The core of the 3D rendering technologies is the Three.js library which provides an abstraction of OpenGL functions to speed up the developments[18].

Figure 29: The scene 3D editor for authoring the scenes of a project

- The upper left panel provides the available assets from a MediaVerse node. These can be dragged and dropped in the scene multiple times, so that multiple instances can exist of the same asset.
- The bottom left panel provides the list of available scenes in the current project. Scene 1 is named as Chapter 1. More scenes with custom names can be created.

[18] Three.js: https://threejs.org/

- The right panel provides widgets for editing the scene such as:
  - Moving, Rotating, and Scaling objects
  - A list of assets instances in the scene that can be used for selecting or deleting them.
- The upper panel provides standard assets for drag and dropping in the scene such as:
  - types of lights: Directional (like Sun), Spot, Point (like a lamp), and Ambient lights;
  - Defining certain positions for actors in the scene as shown in Figure 30.
- The COMPILE button on the upper right corner leads to the compiling of the scene into a multiplaying experience.

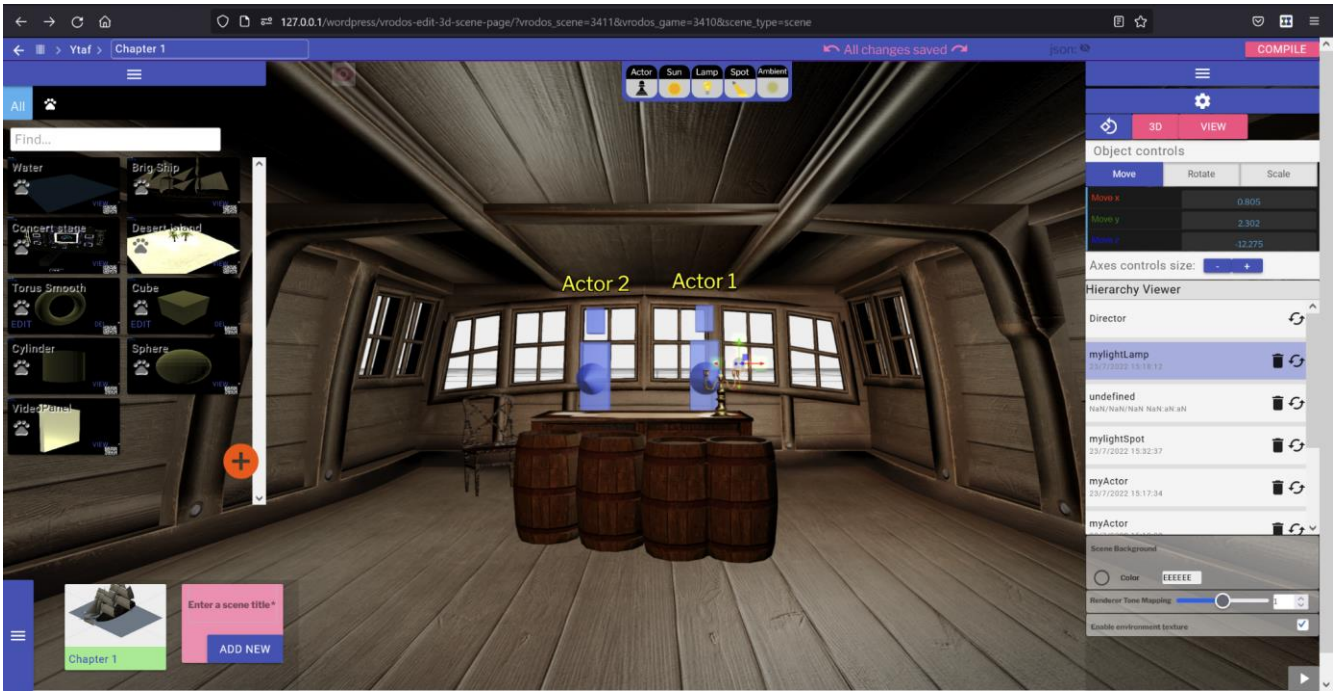

Figure 30: The scene editor allows placing Actors into their position so as to avoid navigational stress during acting

Compiling overview

All the metadata of the scene and the scene itself, with its contents, are saved in a JSON format that is stored inside the WordPress database. This is done automatically every time the user modifies the scene. Although this JSON format is enough for importing and exporting the scene into Three.js, it is not enough for multiplaying experiences. This is due to the fact that Three.js is a framework for 3D graphics, and therefore, it does not provide networking and replication activities for multiplaying experiences.

The most suitable framework for multiplaying experiences is the Networked-Aframe[19] which is based on Aframe[20] and Open-EasyRTC[21] technologies, and they are suitable due to the fact that they are also web technologies and thus can be connected to our WordPress plugin. Aframe is an HTML tags based interface for Three.js. It is related to Three.js, but it operates in a more abstract way. Open-EasyRTC is an abstraction of WebRTC for transferring streams of data between peers such as the affine transformations of the players as well

[19] Networked-Aframe: https://www.npmjs.com/package/networked-aframe
[20] Aframe: https://aframe.io/
[21] Open-EasyRTC: https://github.com/open-easyrtc/open-easyrtc

as their audiovisual channels. However, Open-EasyRTC is a javascript API that uses Node.js server and requires the additional installation of Node.js server. When the user presses compile, then:

- Node.js server runs Easy-OpenRTC main script and exposes an interface at port 5832;
- The JSON scenes are parsed and transformed into Aframe HTML tags. A compiler kernel is implemented using the PHP language as it is shown in Figure 31.
- The compiler kernel transforms the JSON text into JSON objects, iterates over JSON objects, and generates HTML DOM objects accordingly. Each property of the JSON object is transformed into an attribute in the HTML DOM object. Aframe has its own HTML DOM tags that are for example a-entity, a-light, a-ocean etc. The example demonstrates how a JSON object that corresponds to a GLB model is transformed to an Aframe "a-entity".
- Two distinguished web pages are generated that provide the UIs, namely one for the Director with all necessary operations capabilities and one for the actor with very basic interfaces. These are described in detail in the Playing Phase (Section 5.1.3).

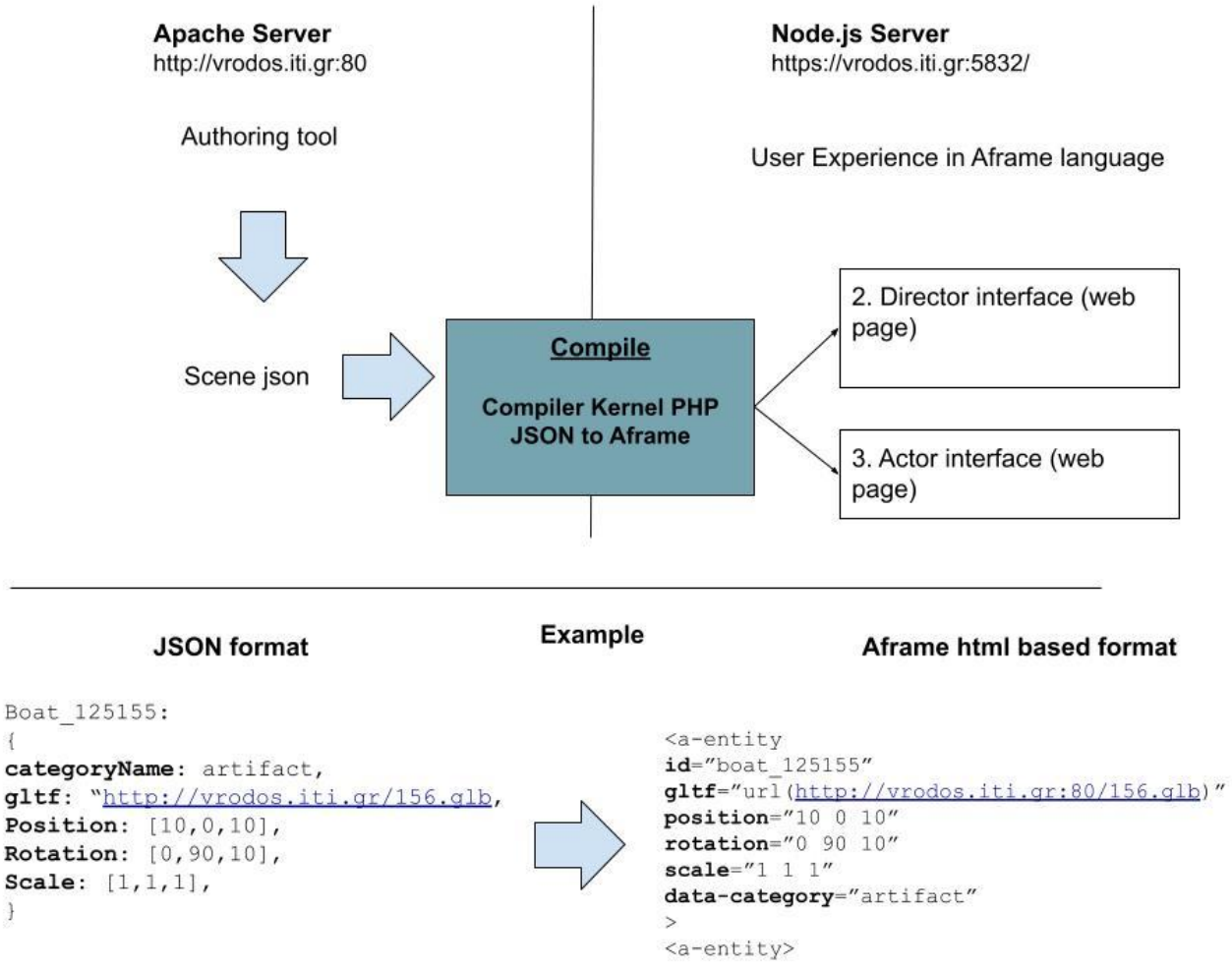

Figure 31: Diagram of the A-frame scene compiling phase

5.1.3 Playing Phase

After the compiling has finished, the UI is shown (see Figure 32).

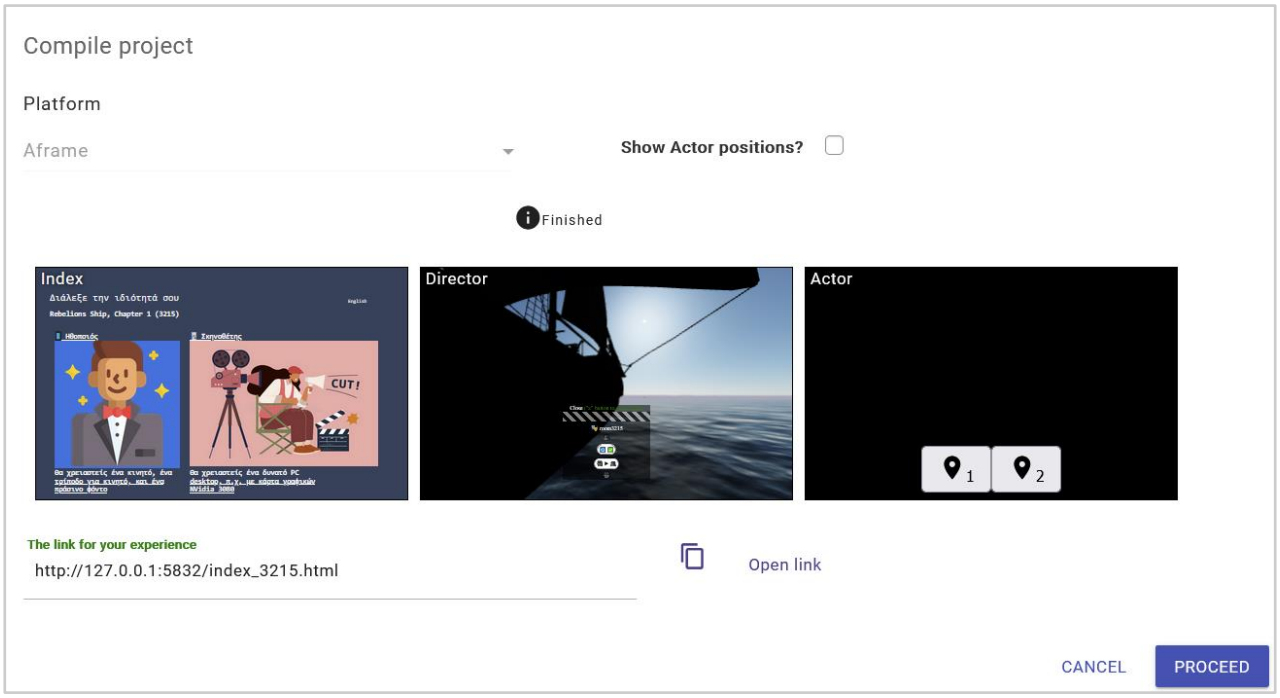

Figure 32: Compiling the JSON into an Aframe multiplaying application

The compiling process generates three html pages, namely:

1. The “Index” page which provides the interface for selecting between the two roles (e.g., Director vs Actor).
2. The “Director” page which provides the interfaces for the Director.
3. The “Actor” page which provides the interfaces for the Actor.

The first link (Index.html) is the only one needed as it leads to page 2 and 3. Each index has an identifier for the scene at its end, e.g. for scene 3215, the link is “index_3215.html”. As shown, the link is related to the server that runs at port 5832. This is where Node.js server is located.

Upon pressing “Open Link”, the index is loaded in a separate tab as shown in Figure 33.

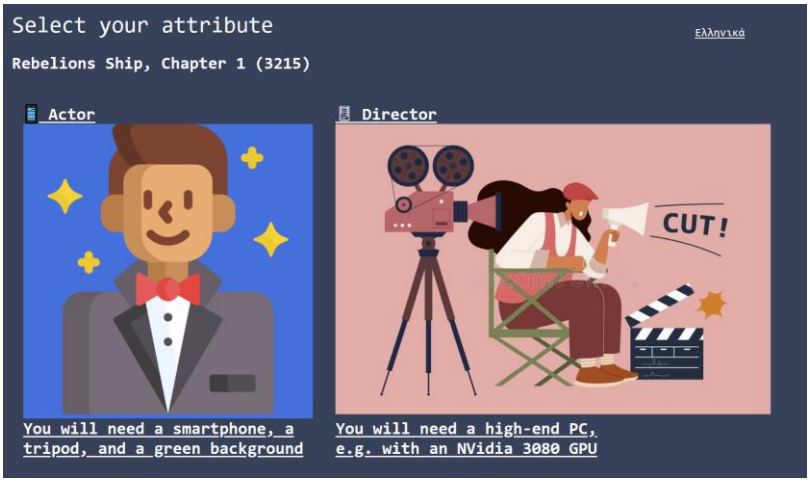

Figure 33: The first screen regards the selection of the user role, namely actor or director. Actors can participate through their mobile devices. Directors should have a high-end PC.

It consists of two choices regarding user attributes, namely (a) Actor or (b) Director. Simple instructions are provided such as what equipment will be necessary for each case. Many actors can participate but only one director can participate which she or he is responsible for the orchestration of the virtual production. The logic followed is the same as a physical theatrical play. On the upper-left side, the project and scene names are shown. On the right-side, the language of the interface can change between English and Greek. The "Actor" and "Director" images are buttons that lead to the following interfaces.

Director Interface

Figure 34 shows the interface of the Director. It consists mainly of the 3D canvas with the loaded 3D models, the lights, and the water element if any. Director can fly wherever in the scene with buttons WASD in order to set the camera to the proper angle and position that will shoot the scene. The water has been implemented using a shader[22] as it has a better quality than using a plain 3D model. A sun-sky component was used for realistic domes[23]. The position of the directional light in the authoring tool was used to set the sun position in the dome.

Figure 34: Director views the scene with all of its properties

An important 2D interface for the director is the "Clacket" as shown in detail in Figure 35.

[22] A-water for Aframe: https://github.com/Dante83/a-water

[23] A-sun-sky for Aframe: https://www.npmjs.com/package/aframe-sun-sky

Figure 35: The Director's clacket provides basic interfaces for the director to orchestrate the scene

It contains:

- **"Close" button:** hides all UIs and makes the web browser in full screen mode as shown in Figure 36, so that it is ready for streaming.

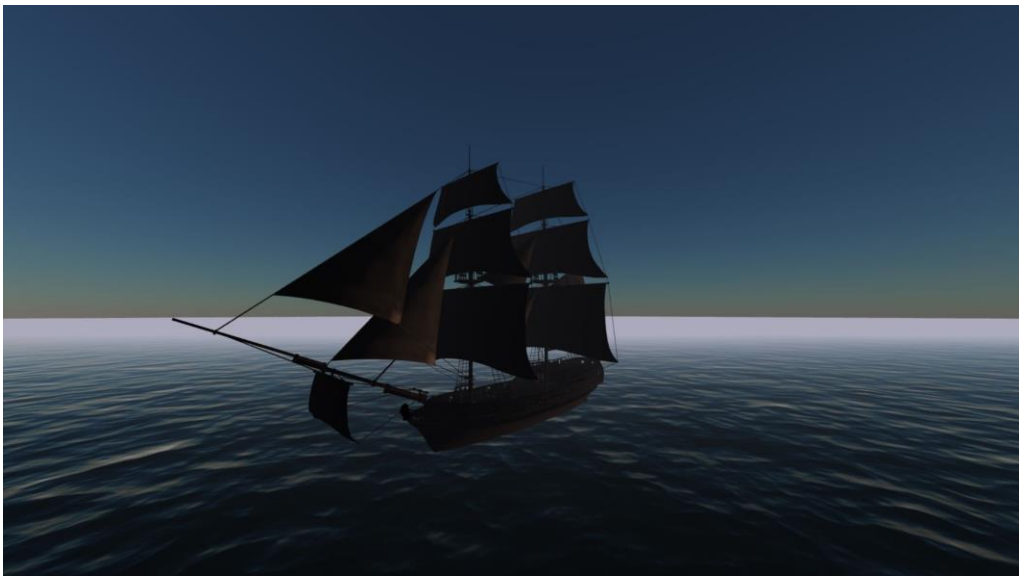

Figure 36: Full screen mode eliminates any UI controls so that screen can be streamed

- **Room id**: The room3215 is an id of the multiplaying room entity. For the sake of simplicity we have one room per scene. Networked-Aframe gives the capability to have multiple rooms that define different spaces. In multiplaying terminology the rooms are called also "Lobbies".
- **"Panels refresh-check" button**  : It shows the UIs for the manipulation of the green screen panels of the actors as shown in Figure 37.

Figure 37: Director has a control panel with several parameters for optimising an Actor's video incoming stream

Whenever a new actor enters the scene, we should press this button so that it is updated with all the actors. A maximum of 20 actors is feasible to participate. To elaborate, when all actors have entered, the Director presses the "Panels refresh-check" button, so that a control interface is created for each actor. The purpose of these controls is to remove the background of the actor's selfie but leave only the actor's body, as well as to make all actors to have the same size. Figure 38 shows the interfaces that remove the background of the user, so that only his body is streamed.

Figure 38: Using the configuration panels for refining green screen thresholds

The methodology to remove the background is based on shaders and thresholds on the amount of green in the scene. Instead of RGB space, we have used the YUV space, which is very often used in green screen composite video background removal. This shader can be found in the Appendix. It stems from a well-known website for experimenting with shaders[24]. The configurations to be done are: i) red, green, blue levels that define the colour of the background, and ii) Threshold Min and Threshold Max that define keep, intermediate, and cut-off levels which are applied to each pixel. Intermediate level is a level of transparency, which avoids pixelation in the edges of the actor envelope. Using all these configuration parameters, the director can optimise the input of each actor since not all actors have the same lighting and background conditions. Apart from the Chroma key method for background removal that we use in this deliverable, we have extensively experimented with AI methods for automatic background removal without green screen. One of these methods is MediaPipe[25]. However, the web version of MediaPipe was found unstable as it was made by transpilling the original C++ code into WebAssembly code. Only in 30% of the cases the users were able to start the experience, in the rest 70% the smartphone was

[24] Shadertoy green screen example: https://www.shadertoy.com/view/XsfGzn

[25] MediaPipe, https://google.github.io/mediapipe/

jammed. In addition, it had random artefacts in selfie segmentation, supported only Chrome browser, and it was computational heavy for low-end devices. Therefore, we opted for the conventional Chroma key method (e.g., by placing a green sheet behind the user), which significantly simplifies the selfie segmentation procedure.

The parameters w and h refer to the width and height of the 2D panel of the Actor stream so as to fit to the perspective of the scene. These are controls on the size of the panel of each actor (i.e., the width and the height). These are used in order to change the size stream panel of the user depending on his distance from his or her smartphone. In this manner, all actors have the same size in the scene.

- The "Camera to actors" button  streams back the Director screen to the Actors smartphones so that they can see the rendered scene (see Figure 39).

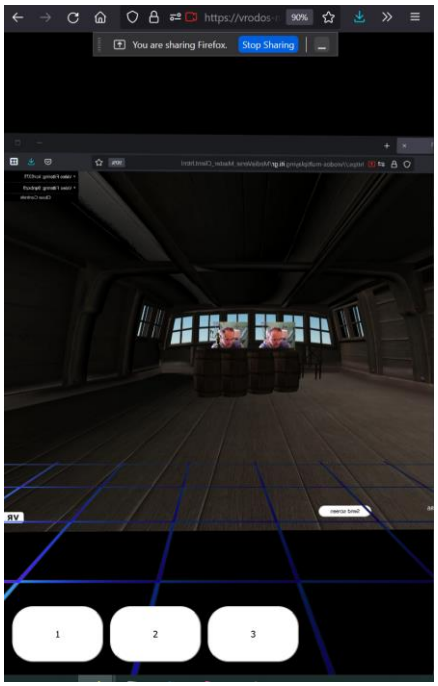

Figure 39: Streaming back the rendered scene in the Actor's smartphone. Buttons 1, 2, and 3 change the position of the actor in the scene. A 2D panel shows the rendered scene

Actor Interface

Figure 39 shows the Actors' interface. It is practically a blank 3D space that contains some location buttons namely 1, 2, and 3. This is for 2 reasons: a) to make it as lightweight as possible (i.e., to avoid rendering any 3D models that will make the smartphone hot over time and exhaust the battery); b) to provide automatic 3D navigation into certain positions where the actor can start acting so as to avoid navigational stress. Feedback to the actor for the current stage status is provided when the director starts sharing his or her screen.

To summarise, the interface of the Actor contains:

1. buttons 1, 2, and 3 to automatically navigate in certain positions in the 3d space (these locations are customizable from the authoring tool).
2. the stage feedback stream (that records the whole stage) as a stream coming from the Director. The Director is the master-client that performs the rendering and submits the rendered stream to all Actors

so as for them to understand that there is action. No heavy 3d model rendering is performed on the actor's smartphone because it is not possible to stress with realistic graphics on their devices.

3. The selfie camera recording the actor's face or full body. Green or blue sheets on the background should be used for each Actor.

5.1.4 Exploitation Phase

The Director can record his or her screen with a screen recording tool and upload it to a MediaVerse node. Another option is to stream live his or her desktop with Facebook live events as shown in Figure 40.

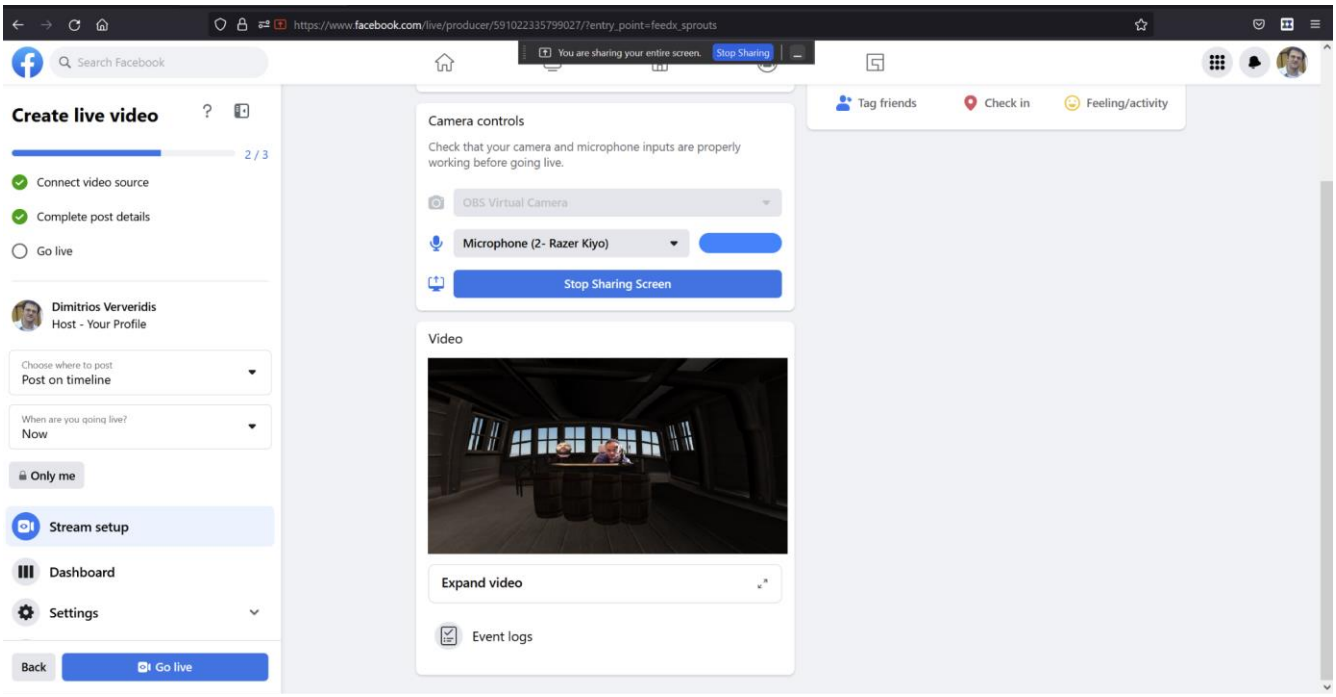

Figure 40: Streaming web browser on Facebook as a live production

In conclusion, our "co-creation" application for remote live performance cases can lead to low-cost high-quality VR media productions. We are innovating by allowing actors to participate in the same 3D experience remotely, live, and using low-cost equipment, namely low-end smartphones. The final product is a virtual production video. The target is to allow the remote engagement of actors due to cost, time, and health restrictions.

5.2 Technical Details for the Multi-playing Capability

Several developments were made on top of the Networked-Aframe software in order to provide the necessary functionalities for green screen sharing in multiplaying scenarios. To recapitulate, we have based our developments on Networked-Aframe (NAF)[26], an open-source and free javascript server software which is based on Open-EasyRTC[27] for peer to peer communication across users in order to increase the Aframe library with multiplaying characteristics such as user position/rotation, user audio, and user video streams. Figure 41 presents the architecture of NAF. It consists of two steps. In step 1 (left), each user (client) is connected to Server

[26] Networked-Aframe, https://github.com/networked-aframe

[27] Open-EasyRTC, https://github.com/open-easyrtc/open-easyrtc

in order to receive the necessary code, namely JavaScript and HTML. One of the clients is the Master client whereas the others are called simple clients. The Master client enters the scene from a different html - JavaScript page than the other clients in order to load the graphics rich scene and have more controls over the incoming streams. However, the room as a shared space is the same. Next, the clients communicate with each other without the need of the server. All the clients have the same role as it is seen on the left as Easy-OpenRTC is a peer-to-peer technology. The benefit of peer-to-peer technology is that the network is not destroyed if one client leaves as all clients send their stream to all clients. The drawback is that the number of nodes increases the traffic, something that is not happening in centralised server topologies.

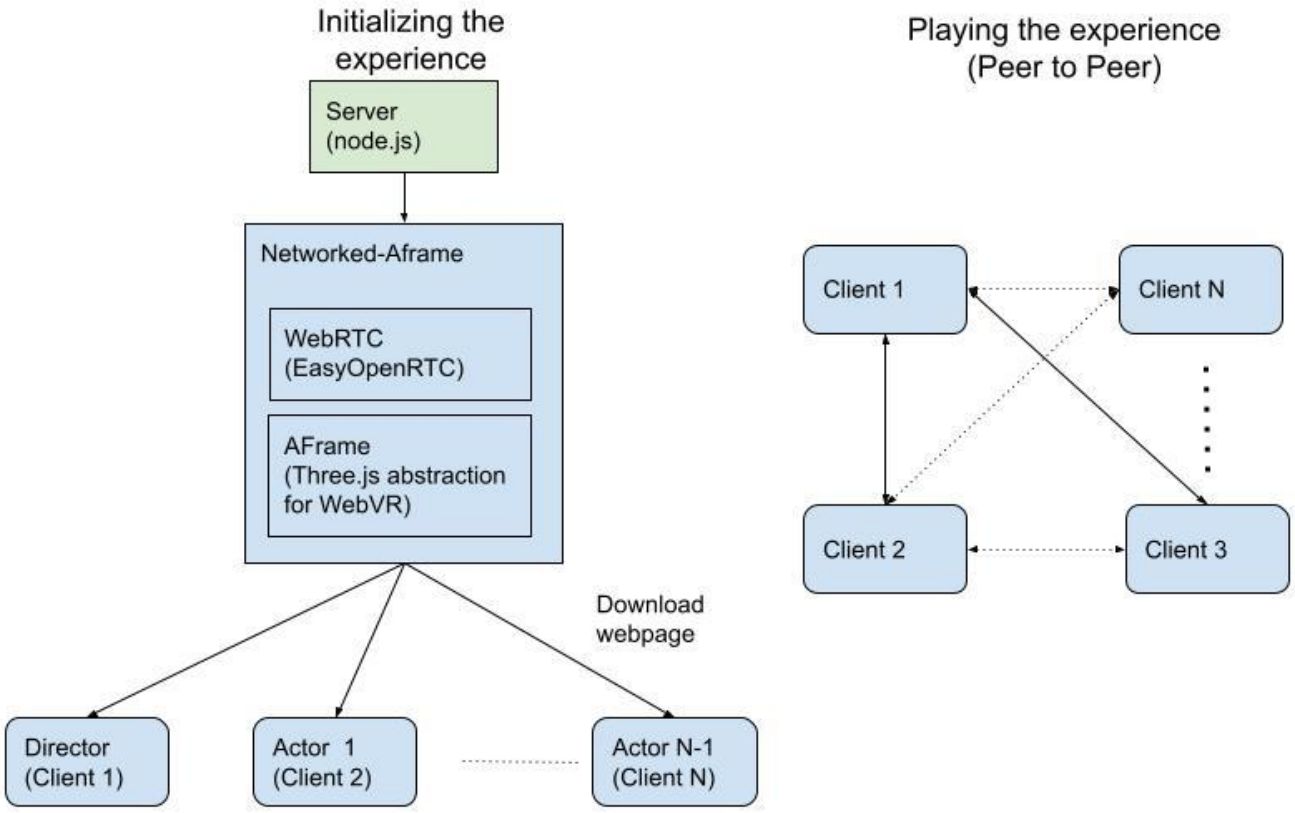

Figure 41: NAF architecture for multi-player VR experiences

Our contribution to NAF is the writing of custom shaders and configuration tools for them on the fly while rendering. These shaders can be found in Appendix I. Shaders are used from the master client when receiving the incoming streams from all clients and replaces on-the-fly the green pixels with transparent ones. The benefit of using Shaders is that they are much faster in rendering than any other processing (e.g. canvas image processing) because they are executed on the GPU. Figure 42 depicts the architecture of this procedure. In our experiments, we have used a high-end PC equipped with an NVidia RTX 3080 graphics card.

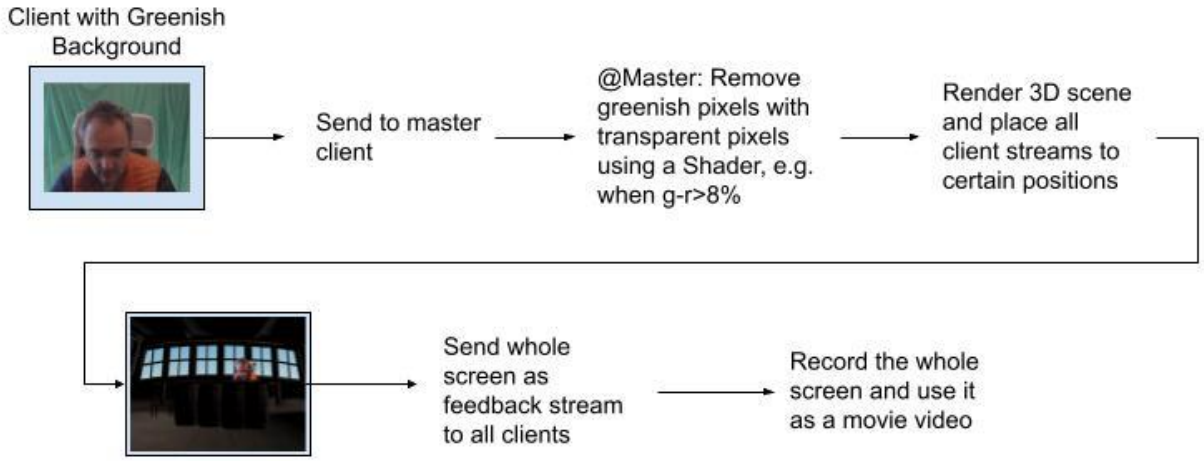

Figure 42: Overall logic when the chroma key is the same among all clients

5.3 Integration with the Main MediaVerse Node

The basic step for the integration of VRodos with the MediaVerse node is authentication with the same credentials. The integration of the authentication scheme between VRodos tool and an MV node relies on the development of a custom plugin in the VRodos web platform that handles the communication between the two components, as presented in Figure 43.

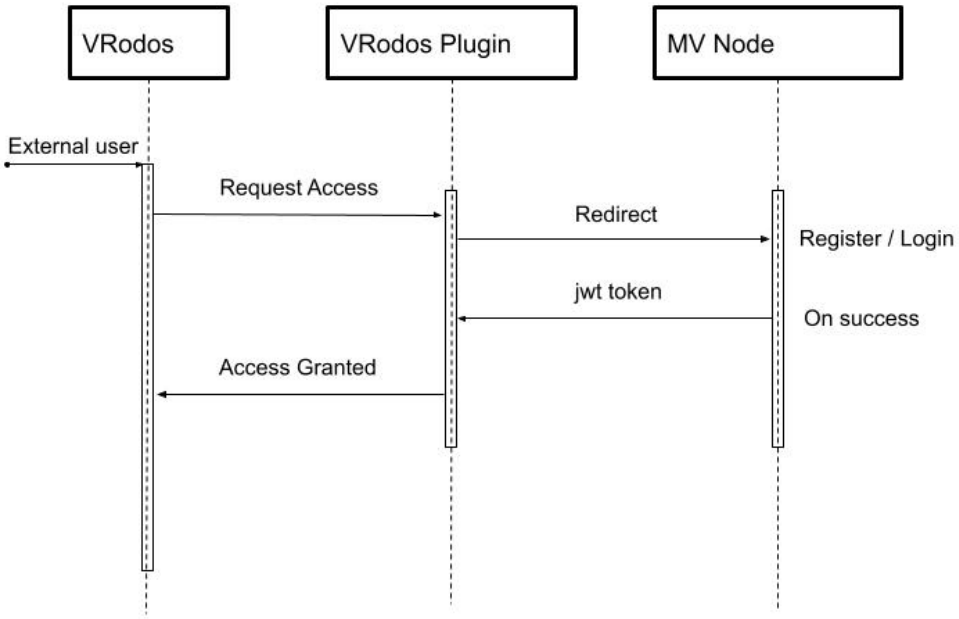

Figure 43: Sequence diagram of the authentication scheme between VRodos tool & an MV node

This procedure consists of the following steps:

1. A user is registered in the MV Node
2. Once a user has registered in the MV Node, he or she can request access using the Login Form from VRodos in order to be authenticated.
3. On success, a user receives a valid JWT (JSON Web token) from the MV Node, which is used from the VRodos plugin in order to allow the user access to the MediaVerse tool assets.

5.4 Addressing MediaVerse Requirements for the VR Collaboration Sandbox

The methodology developed is complementing the VRodos authoring tool presented in D5.2 with appealing functionalities. Both D5.2 and D5.3 are addressing several of MediaVerse requirements[28] such as those described in the following.

XR-01.2 User presence in VR through Webcam

This functionality has been described extensively in Section 2.4, which provided the methodology to represent an avatar by his or her video.

XR-01.3 interfaces for VR devices

The interfaces for VR devices are supported by Networked-Aframe platform, which is an extension of Aframe[29]. They support VR controllers and Head Mounted Displays. We have successfully made templates for VRodos that exploit the Oculus Quest 2 standalone VR headset. More details about the interfaces for VR devices, Smartphones, Tablets, and PCs were described in MediaVerse D5.2 - Immersive Storytelling Authoring Tools v1 [30].

XR-03 Import 3D objects

The VRodos authoring tool has the ability to host 3D models or use 3D models from external sites. The methodology to achieve this interlinking consists of two steps, namely:

1. **Interfaces development:** Developing the popup interfaces in the front-end for the 3D scene editor of VRodos in order for the user to select which model to fetch from MediaVerse 3D repository.
2. **Ajax query for assets' list:** through questioning MediaVerse central node with an ajax query live from VRodos 3D editor in order to get the list of available models in the form of JSON which consists of assets (3d models) title, url link of the GLB, and an icon.
3. **Deep Linking:** through loading the glb and saving it the VRodos scene as JSON entry.

XR-04 Decorate VR walls

Lately, we added the ability on VRodos authoring tool to insert video onto walls as moving textures. The user can right click on a 3D object and select the video that wishes to replace the object texture. In the future, we will also give the ability of the author to assign images. The user can configure the repetition number of the video texture in X and Y axis, as well as the offset of the video from 0,0 of UV texture mapping of the object (Figure 44).

[28] https://gitlab.com/mediaverse/wp6/-/issues?search=XR&sort=created_date&state=all

[29] Aframe, https://aframe.io/

[30]https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D5.2_Immersive-Storytelling-AuthoringTools-v1.pdf

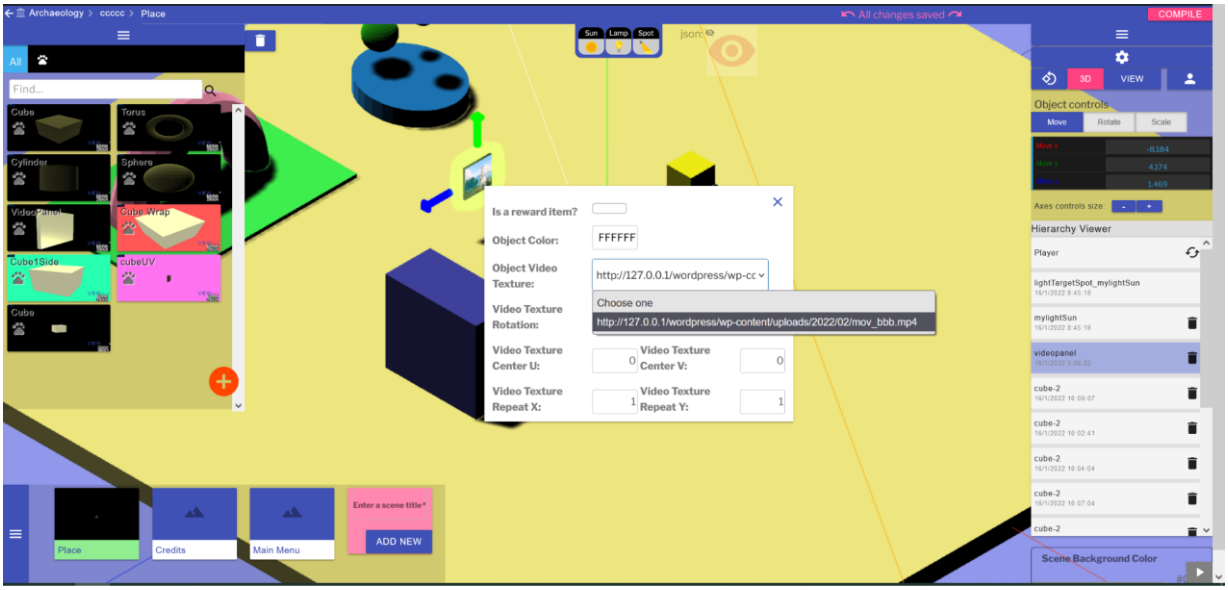

Figure 44: VRodos functionality for changing the texture of an object with a video

5.5 Comparison with Other Methods of Virtual Production

Virtual production has been in the market for a long time but not through web browsers and using real-time streaming as our approach. We compare the proposed tool that we simply call VRodos-NAF, with two other popular virtual production tools namely Blender[31], and Unreal Engine 4[32] that are well known software in the Movie and Gaming industries, respectively. We compare the three tools with respect to the features they offer, the quality in graphics, and the expertise level needed to operate them. In order for the comparison to be achievable, we have used the same 3D model and surroundings, namely the 3D mode of the Brig ship with the water element around it.

5.5.1 Blender

Blender's main use is for designing and rendering 3D scenes using its "Geometry" tab as the one shown in Figure 45. Moreover, Blender offers through its "Composite" tab (see Figure 46) the ability to process video files such as to replace their Chroma key with transparent pixels. In the main tab "Geometry" (see Figure 47), the videos can be assigned to planes and positioned anywhere in the 3D scene. Finally, by pressing "rendering" the scene frames are rendered in coordination with the Chroma video frames into a single video file.

Blender has many visual programming tools and therefore coding is not needed. However, extensive skills in using its GUI are needed. The result is very realistic as it is rendered with ray-tracing methods. This demands that the whole editing and rendering should be done in a high-end PC. For example, the final rendering of the scene of Figure 47 took about 3 secs per frame by exploiting a NVidia RTX 3080. So the overall ratio for 24 frames per second is 72:1 (i.e., a video needs 72 times its actual length in order to be rendered within a 3D scene and exported to another video file). Therefore, it becomes obvious that virtual production is an asynchronous process for experts in Blender.

[31] Blender, https://www.blender.org/

[32] UE4, https://www.unrealengine.com/en-US/

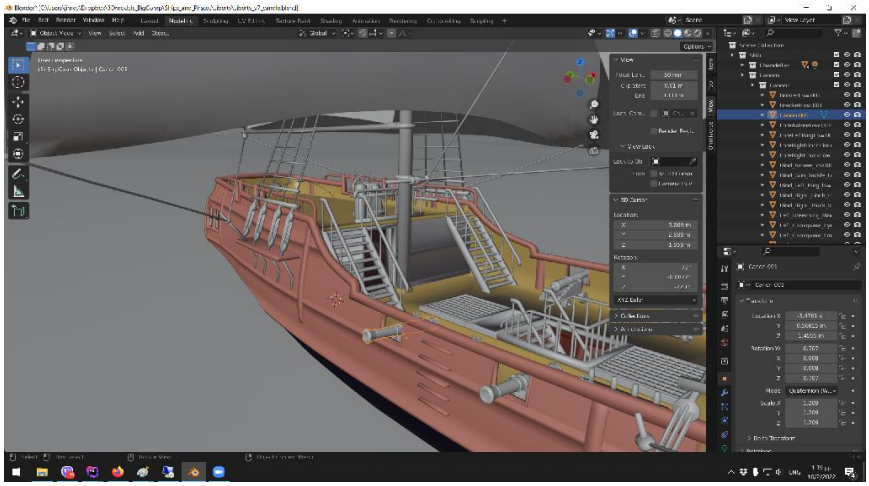

Figure 45: Constructing the scene

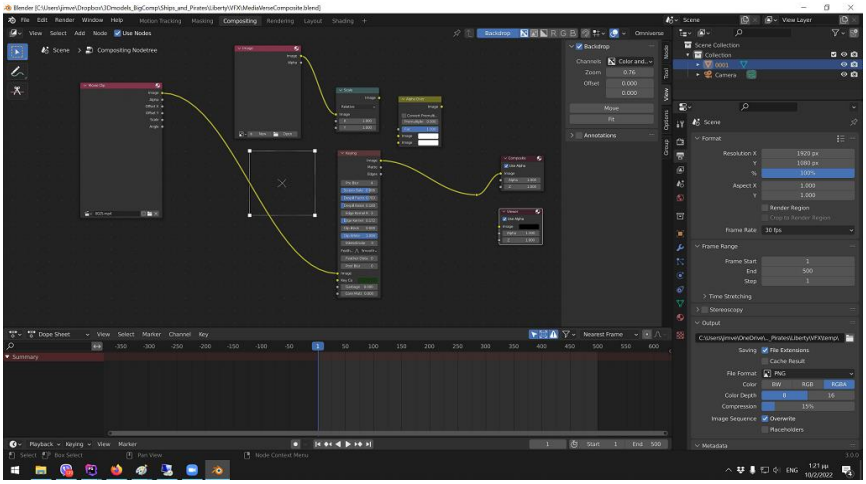

Figure 46: Removing Video Chroma Key

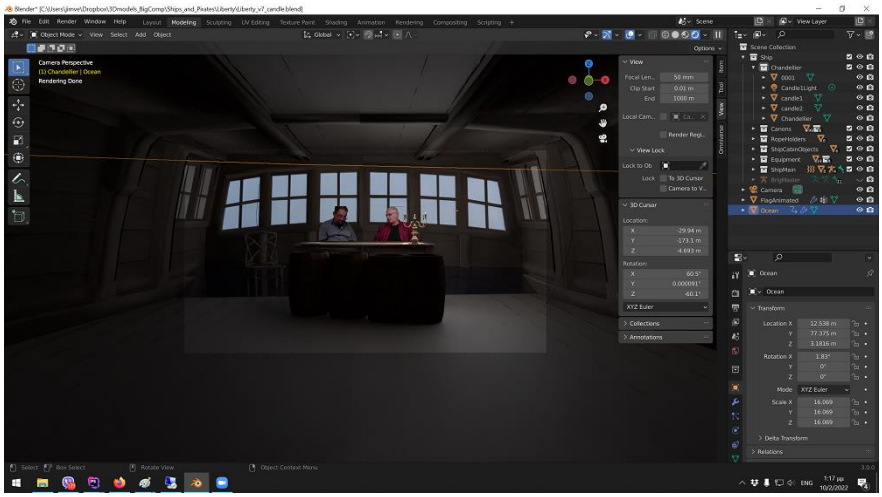

Figure 47: Combining 3D and video

5.5.2 Unreal Engine 4

UE4 is a graphics engine that is being extensively used for virtual production by many studios. Using a video camera with a tracking device the directors can capture actors in a Chroma screen studio as well as the position/angle from which the shot was taken to replace the Chroma key with the 3D scene rendered from the correct position and angle. Another way is to use huge screens instead of Chroma key background where the scene is rendered on the background from the correct angle taken from camera position/rotation tracking.

UE4 could be potentially used for remote virtual productions instead of the proposed method based solely on Web technologies. UE4 supports multi-user existence on the same scene, chroma key removal on videos, more realistic graphics, a visual programming interface, and remote rendering through its Pixel Streaming service, which is accessible by a Web interface for all clients. Additionally, it has many add-ons such as the Cesium 3D geographic maps that can increase the realism of the background.

Its cons is that it is not directly compatible with web technologies such as the VRodos authoring tool so that the authoring interface should be developed from scratch. UE4 could be potentially used in a future approach if the web technologies are not sufficient for rendering realistic sceneries, producing thus non acceptable quality results. In such a case, another 3D editor should be made as a standalone application which should be installed in user smartphones. In Figure 48, we have set up a scene with the same graphics elements in order to demonstrate the quality achieved by UE4. It is almost as good as Blender non-real time rendering, but superior to WebGL technologies. Of course, UE4 can be also used for non-real time rendering and produces results similar to Blender.

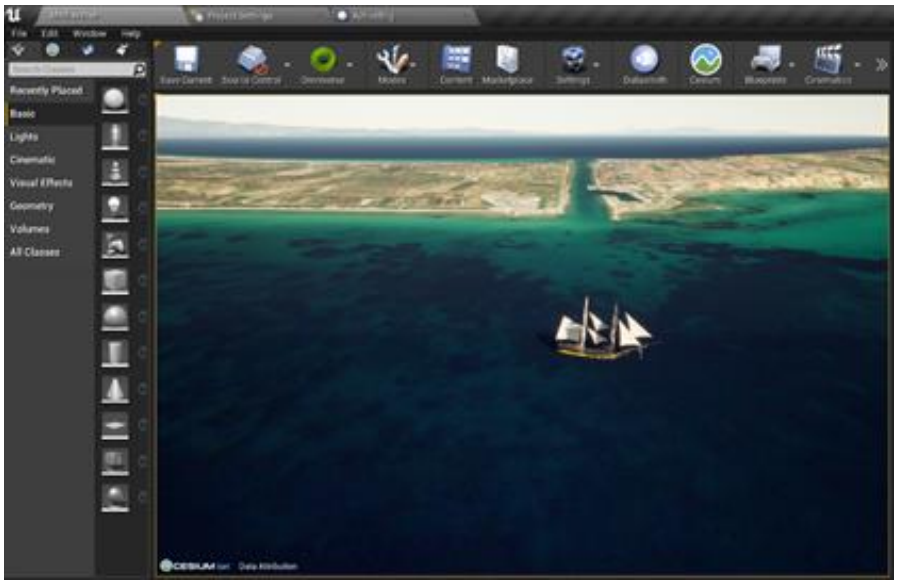

Figure 48: Scene in UE4 rendered in gaming mode (real-time)

5.5.3 Comparison of the Main Features

The proposed method VRodos is compared to Blender and UE4 methods for virtual productions across their main characteristics in Table 1. It is seen that the main drawback of VRodos is the Medium quality in 3D graphics as it is based on WebGL, which is much inferior than binary based engines such as Blender and UE4. We have significantly improved the quality of the graphics using shaders. In Figure 49, Figure 50 and Figure 51, we compare

the graphics of each method. Significant issues exist in the reflections and shadows in Three.js. There have been constant improvements in the quality of graphics in the latest years by Three.js so that we hope we can achieve high quality in 2 or 3 years. The advantage of VRodos vs Blender is the multiplaying capability as Blender does not support any multiplaying. The web-based editor is the main novelty of VRodos, which currently Blender and UE4 do not have.

Table 1: Comparison of VRodos-NAF with Blender and UE4

	VRODOS	BLENDER	UE4
Realism	Mid	High	High
Multi-user collaboration	Yes	No	Yes
Pixel streaming	Yes	No	Yes
Licence	Apache 2.0 - MIT	Blender	Proprietary
Editor for non experts	Yes	No	No
Programming need	No	No	No if Blueprints are used
3D editor	Yes	Yes	Yes
Language for programming	Javascript	Python	C++
Other	Aframe and three.js are supported by Mozilla and Google respectively. Many add-ons, active community.	Supported by many funds. Many add-ons, active community.	Most realistic graphics engine for games. Many add-ons. Marketplace with professional tools. Analytics support.
Main industry	Web Publishing	Movies	Gaming

Figure 49: VRodos rendering example

Figure 50: Blender rendering example

Figure 51: UE4 rendering example

6 Social Analytics for Content Enriching and Performance Tracking

Regarding social analytics, we have focused on the following three tasks, namely:

1. The content ingestion from owned social media accounts for the enriching of a MediaVerse node content;
2. Content publishing in social media platforms; and
3. Performance tracking of shared content, including trending topic detection.

6.1 Setting up Personal Social Media Accounts

Authentication through Firebase is used to allow users to sign in to personal social media accounts and to be able to interact with the social media platforms directly from the MediaVerse node. In this deliverable, we deal with the first step towards content ingestion for the enriching of media content, which is the access to the information. We have implemented the OAuth mechanism in order to retrieve the authorization tokens from the relevant social network in order to manage the User Generated Content (UGC) belonging to the authenticated user. This allows the user to retrieve and store the relevant content inside the MediaVerse Node storage. We have focused on Twitter login but the same techniques can be extended to other social media like Facebook, Google, etc. We have implemented the OAuth handshake with the usage of Firebase, which offers a wide set of facilitators for the social login. Firebase Authentication is used to allow users to sign in to personal social media accounts and be able to interact with the social media platforms directly from the MediaVerse node.

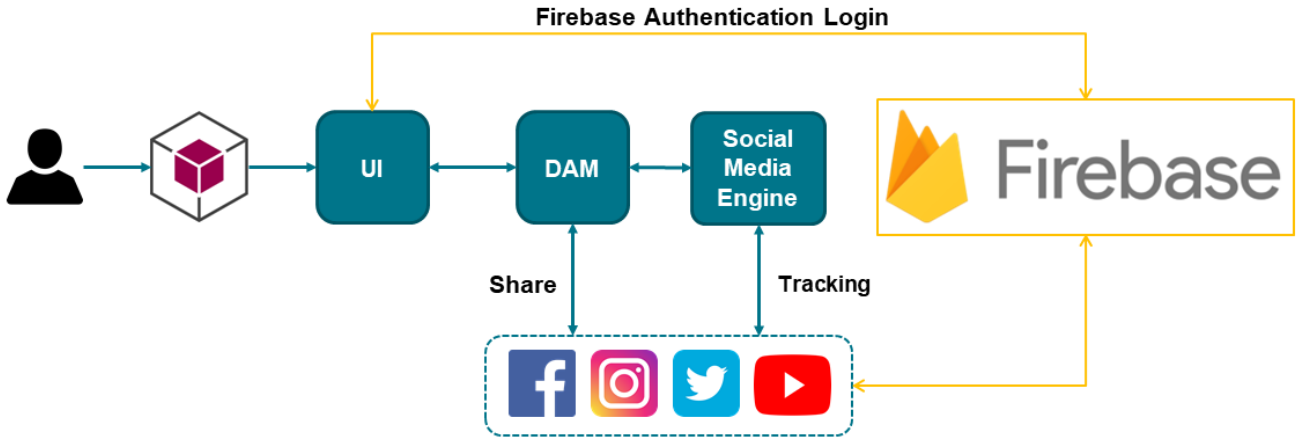

Figure 52: Social analytics architecture diagram

The whole process is handled via the most common library for node package manager named Firebase[33]. It contains documented functions with several functionalities, eliminating the need to implement them from scratch. For the interconnection between the UI and the Firebase we have created a new app to store the project ID and the web API Key provided by Firebase. These two variables as shown in Figure 53, must be defined during the deployment of the MV node as described in the official repository of the project. Towards this goal, Firebase JavaScript SDK was installed, configured, and it is part of the UI module of the platform.

[33] Firebase - https://www.npmjs.com/package/firebase

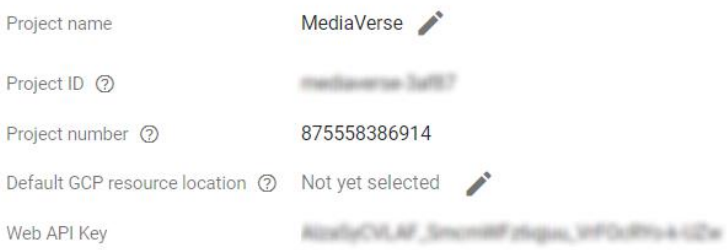

Figure 53: A new Firebase app contains the necessary parameters for the authentication of MediaVerse and Twitter APIs

The next step is to register a Twitter provider on the Firebase project as shown in Figure 54. Several other providers can be used, such as Facebook and Google, as mentioned in the previous paragraph.

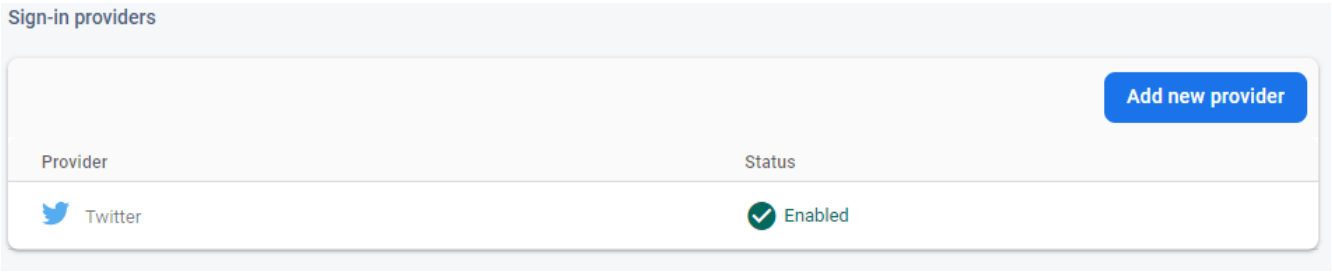

Figure 54: Create Twitter as a new provider of data in Firebase

In order to do that, we have to provide a valid API Key and API secret in the Firebase Sign-in settings. Those two keys can be found in the Twitter developer portal. A callback URL (which Twitter uses for redirecting the user after a successful login attempt through a pop-up window) must also be set in the Twitter developer portal (auto-generated by the Firebase sign-in provider). Those three parameters are necessary for the interaction between Firebase and Twitter and must be the same in the two platforms.

Figure 55: API Key and Secret configured in the Twitter developer portal

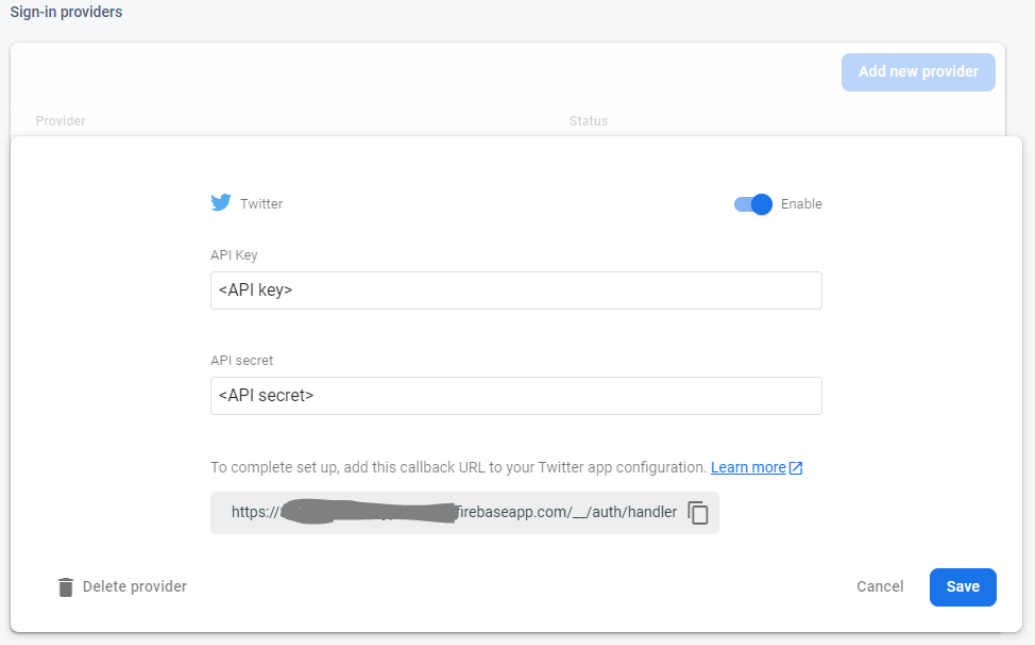

Figure 56: Configuring the Twitter provider in the firebase

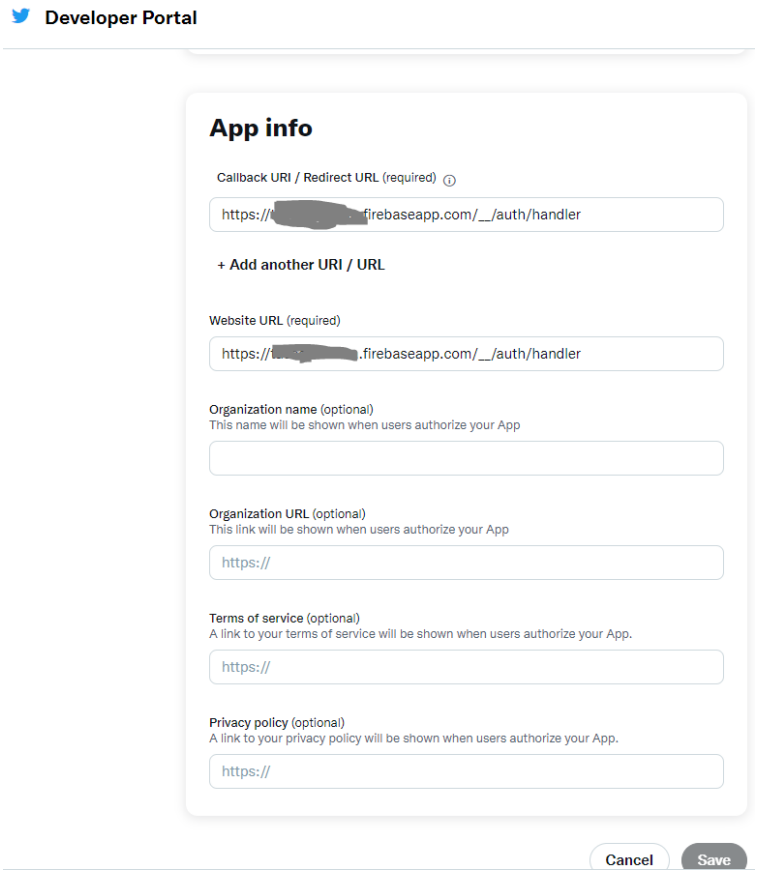

Figure 57: Configuring the callback URL in the Twitter Developer Portal

After the basic settings, the app is ready to communicate with the Twitter OAuth 2.0 authorization server and to handle some core for Twitter functionalities like login, logout, access tokens, user information, etc. The Firebase software is triggered whenever a user wishes to sign in to his/her own social media account.

A pop-up window is shown where the user is prompted to provide his/her Username and Password as shown in Figure 58, which is used in the Social Media platform. Firebase will take care of the OAuth sign-in procedure and acts as a proxy between the user and the social media authentication mechanism.

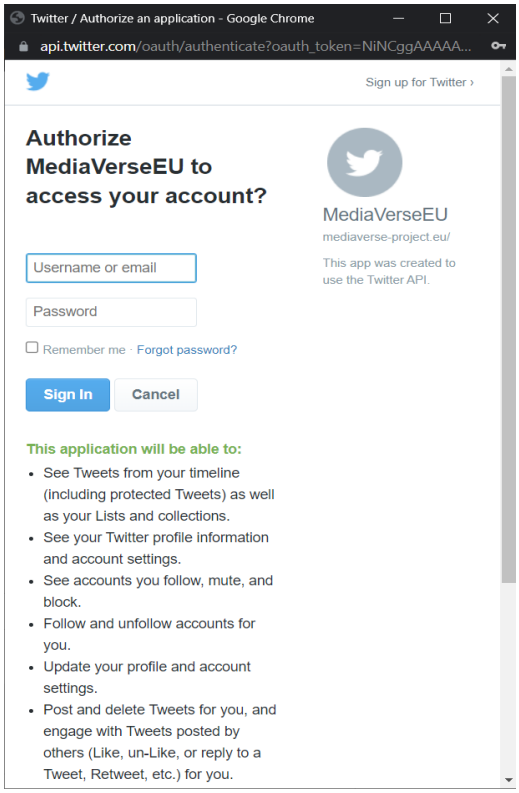

Figure 58: User provides the credentials to sign in to the social media platform

When the sign-in procedure is completed successfully, the credentials of the user are exchanged for OAuth tokens generated by the social media platform. Those tokens will be used by the MediaVerse node for interacting with the social media platform on behalf of the user.

After a successful login, Twitter provides the app with all the necessary information about the user such as "Username, Alias, Thumbnail, Profile URL" and the necessary tokens which will be used for uploading assets on the MediaVerse node on the back end side. One of the biggest advantages using Firebase is that the application can keep tracking of the connection for the user. The Firebase library in the background is using local storage for storing all the necessary information about the user. It means that after closing the web browser the user is still connected and he/she will continue to be connected unless she clears the browser or she chooses to press the logout from Twitter button. In that case, the Firebase cache will be completely deleted and the flow will start from the beginning if he tries to reconnect again with Twitter. Another advantage for Firebase is that it keeps the user logins at logs. Therefore, we keep tracking the users inside our application and export more analytics for further research.

Authentication through Firebase is used to allow users to sign in to personal social media accounts and to be able to interact with the social media platforms directly from the MediaVerse node as shown in Figure 59.

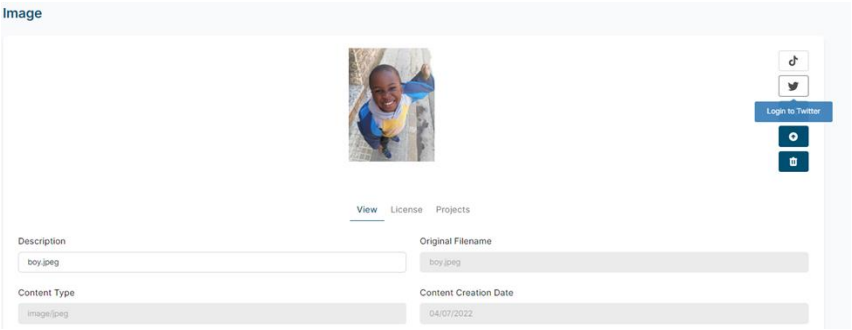

Figure 59: Access to Twitter content through the MediaVerse platform

6.2 Content Publishing and Performance Tracking

The user is able to publish content on various social media given the authorization tokens retrieved with the social login as it was described in Section 6.1. This task also involves performance tracking of the impact of the specific item (e.g., number of likes, number of retweets, number of mentions, etc.). For this specific task, we have improved our existing algorithms to understand and measure the propagation of a specific asset or topic published by an MV user throughout the social media community.

Content publishing

The publishing of an asset is taking place on the Digital Asset Management (DAM) API. After a successful login, the UI is responsible for providing the access token and the secret key on the back end on each post.

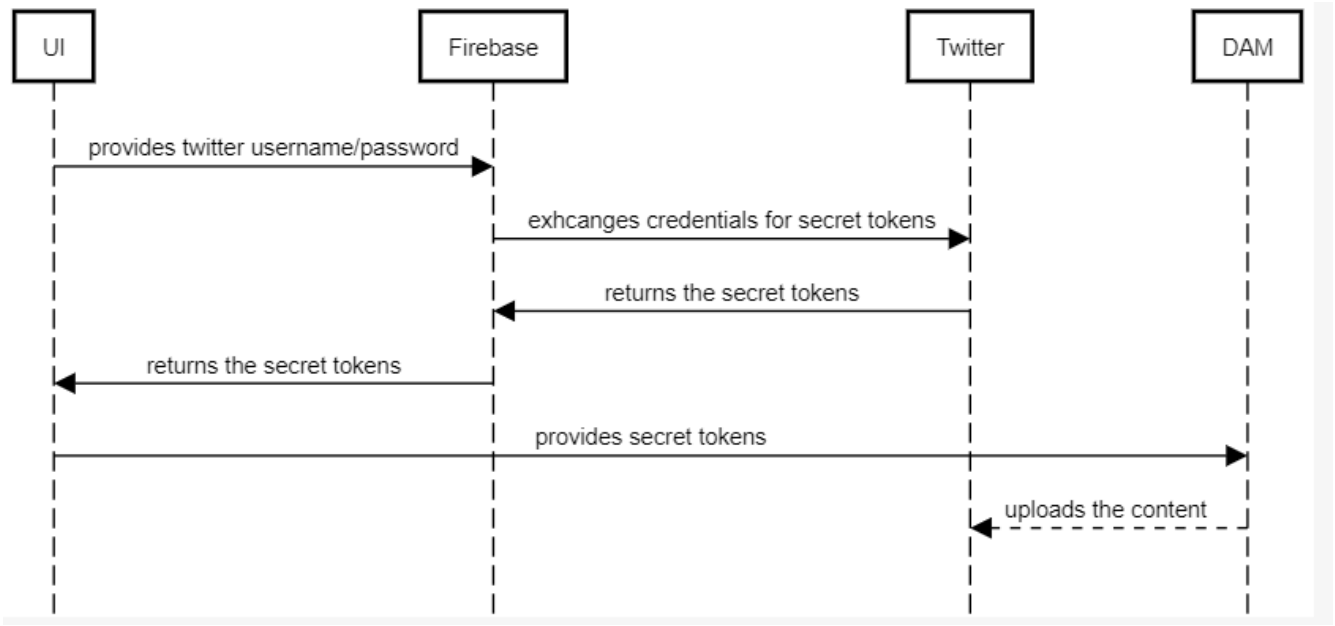

Figure 60: DAM uploads the content with the secret keys from UI

If the post is successful, the uploaded asset should be revealed on her or his Twitter account as shown in Figure 61 and 62 for an image post, and in Figure 63 and Figure 64 for a video post.

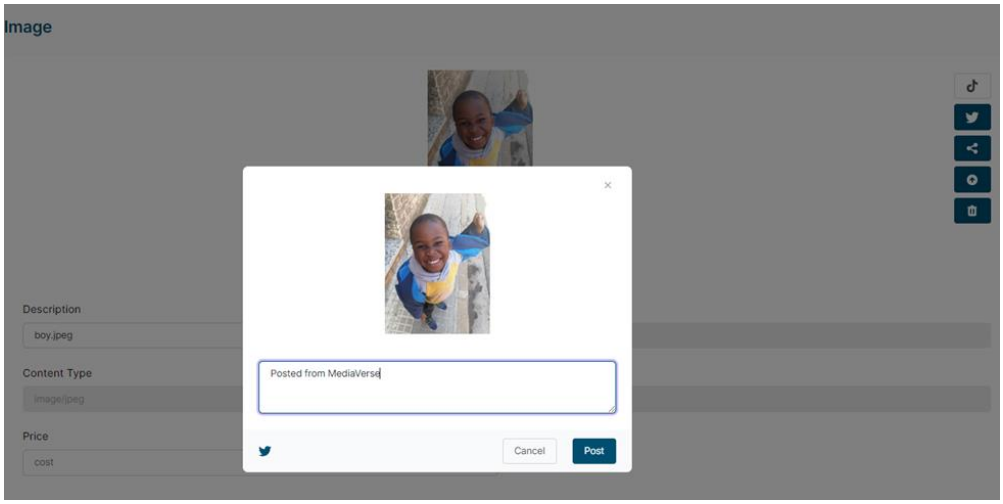

Figure 61: User uploads image to Twitter directly from the MediaVerse dashboard

Figure 62: Image uploaded to a Twitter account

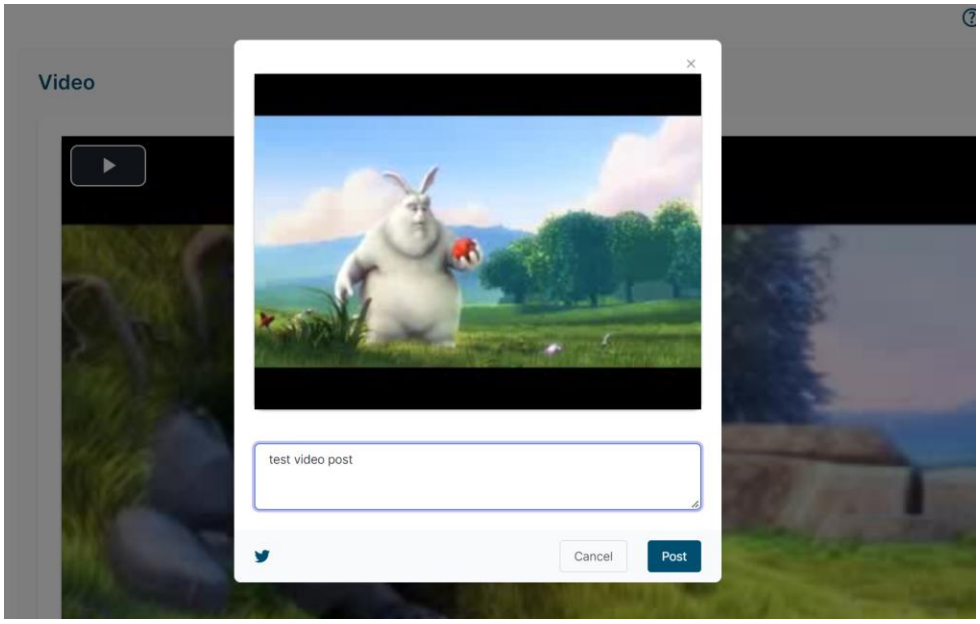

Figure 63: User uploads video to Twitter directly from the MediaVerse dashboard

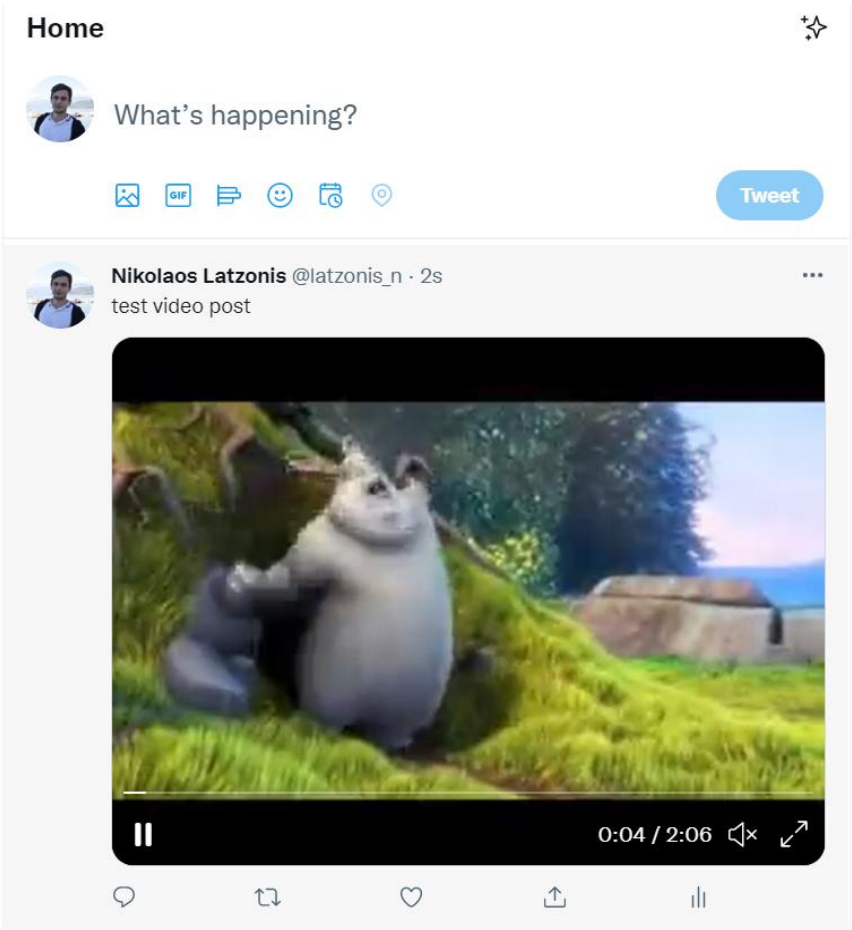

Figure 64: Video uploaded to a Twitter account from MediaVerse node

6.3 Performance Tracking

For media content that is posted on Twitter or in general for MediaVerse users who are interested on monitoring Twitter for trending topics and media, we have decided to extend the functionality of paid APIs - the commercial/marketing APIs of Twitter offer a wide set of KPIs to measure the impact of owned posts - with a set of more in-depth metrics, such as the Social Media reach, the way of propagation of a post as well as the detection of communities discussing about a topic. Overall, the performance tracking is separated to the following activities:

1. Fetching Tweets information;
2. Create the Propagation Graph and calculate Social Media Reach;
3. Detect communities and visualise interactions;
4. Trends detection:
    a. Social Media Monitoring;
    b. Trend Detection;
    c. Topic Detection; and
    d. Influencers - Insights.

6.3.1 Fetching Tweet Information

The Twitter search API allows fetching tweets containing a set of keywords for the last 10 days. Such a scenario refers to the performance tracking of a published URL or the performance tracking of a campaign launched by a user with a specific #hashtag. Our solution is using the Twitter RESTful API and is performing a set of HTTP requests towards Twitter in order to retrieve all the related tweets of the last 10 days.

6.3.2 Create Propagation Graph and Calculate Social Media Reach

The Propagation Graph visualises the evolution of the monitored topic (e.g., a specific #hashtag or URL) by the depicting of the containing tweets in time as shown in the following figure. The X-axis of the diagram is referring to time, while the Y-axis is showing the number of retweets for a specific Tweet. Each circle is symbolising an original Tweet (non-retweet) with circle diameter being proportional to the related author's influence (number of followers for the specific profile). Each circle is clickable so the actual tweet is revealed. The propagation graph can show the reason why a #hashtag has become viral and can be combined with the Social Media Reach diagram, which shows the potential impressions of a Tweet. The latter is calculated by the sum of the number of followers for each retweet involved in the collected dataset and is visualised in an aggregated graph. Figure 65 shows an example of propagation and Social Media reach diagrams, referring to the investigation of a specific topic (a discussion based on the name "Augusto Roux").

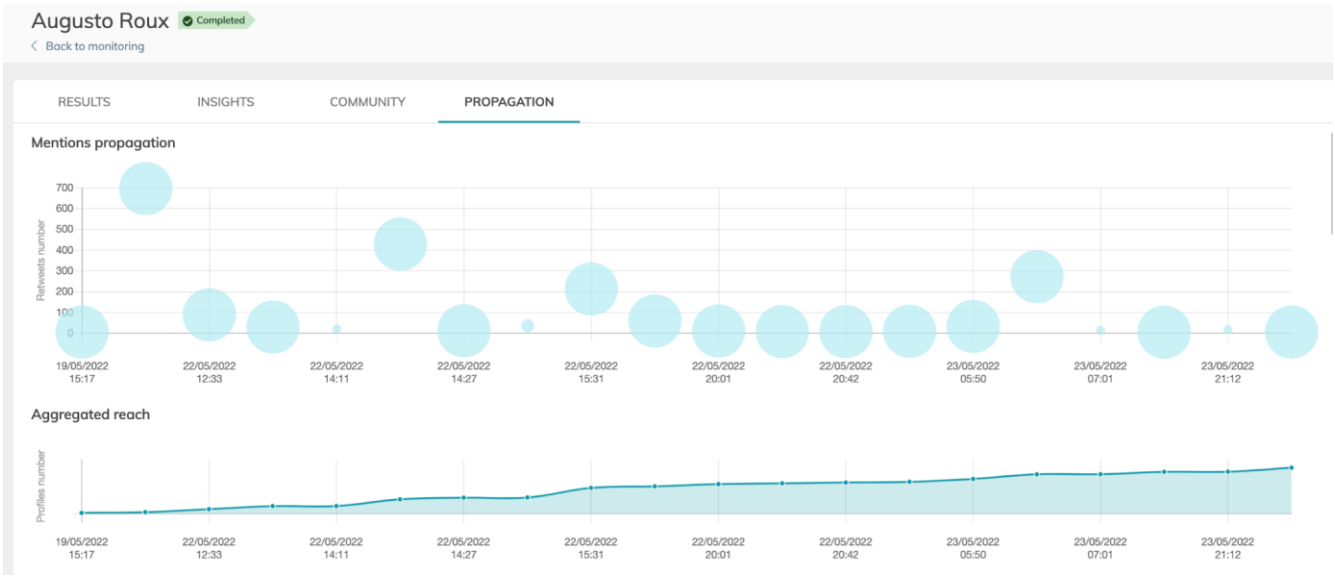

Figure 65: Propagation and Social Media reach graphs

6.3.3 Detect Communities and Visualise Interactions

Another important feature for the monitoring of performance is the detection of the involved communities. When referring to "communities" we are referring to groups of users, which tend to discuss similar topics and usually interact with each other. For this purpose, we have created a clustering algorithm, which retrieves the past activity of each Twitter's profile (e.g., the last 100 tweets and likes) and tries to group them together in the form of clusters. The groups are visualised with the set of keywords describing the interests of the community with the related profiles sorted by influence as shown in Figure 66.

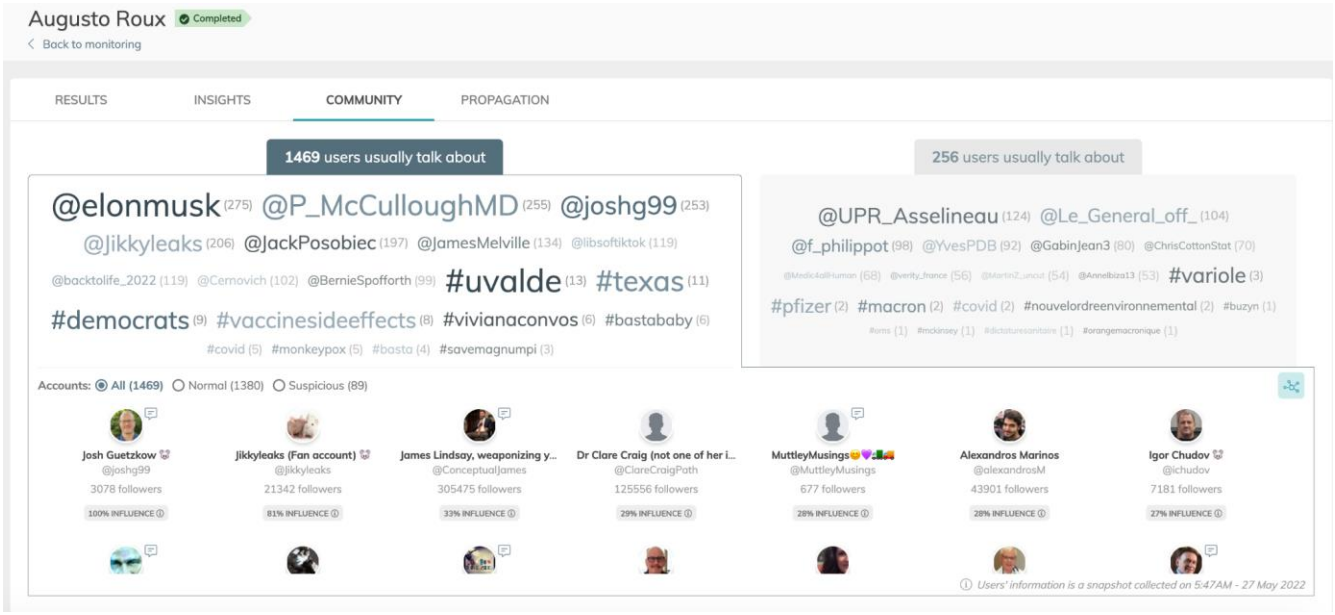

Figure 66: Community description and members

We also offer a network graph visualization with the most influential member including the interaction between them, as shown in Figure 67. Given a graph that models entities as nodes and their relationships as edges, the

community detection algorithm generates random paths starting from each node and visiting its closest neighbours. The sequences generated are then used to train a shallow neural network to predict whether a node is usually found close to another or not. After training the information contained by the graph is embedded in the neural network in the form of the coordinates saved in the weights. Provided that the sequences generated closely describe the graph and that the network has been trained sufficiently, entities close to one another in the embedding space are considered to be closely connected. Finally, a clustering technique is used to sort this space into clusters (communities). The community detection is performed with the use of machine learning algorithms built with Python and TensorFlow.

We can see the two communities around the topic “Augusto Roux”; a French one indicating that there has been the dissemination of this topic in France as well as an American community usually discussing conspiracy theories (e.g. #vaccinesideeffects). It is important to examine the network graph to understand the key players for the distribution of the topic.

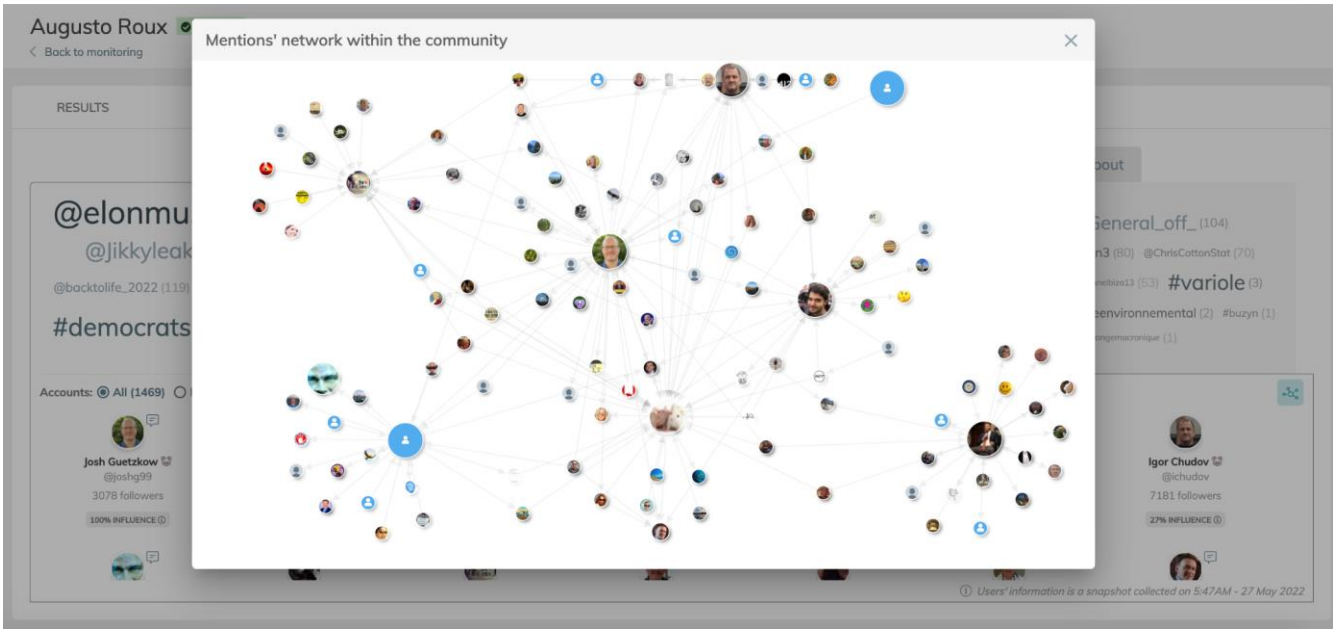

Figure 67: Network Graph of the community

6.3.4 Trends Detection

For the implementation of this task, we have extended the platform Truly Media with Social Media listening functionality. This functionality is executed with the collaboration of the following services/modules.

Social Media Monitoring

The Content Aggregator service provides an effective search, based on the content and metadata information. This service is able to gather content from any online channel and news RSS feeds. Apart from RSS feeds, this module will be able to aggregate and index structured data from various sources, structured news portals, blogs and social media channels (Facebook/Instagram public pages, Twitter). The purpose of the module is to reduce the time and effort needed to check regularly different sources for updates, providing the users a unique environment for monitoring and searching information coming from different online providers. A multithreaded task is responsible to periodically check and fetch latest news/posts from a predefined list of urls or social media

channels according to specified criteria. The administrator of the system configures the list of urls and the scheduler. Logging is available in order to help administrators to diagnose issues.

Aggregation from social media can be based on a number of "social media streams" combining properties like hashtags, keywords, authors, etc. In order to support efficiency in search, the items fetched are indexed appropriately, including both the textual content and any available extracted metadata. In this way, advanced search will be available, including combination of words and proximity search in any supported language. Users are also able to search with free text queries and filter the results by selecting entities (like persons, locations, organisations, etc.), which have been automatically recognized.

The storage module of the content aggregation layer, which contains all the fetched documents is stored in a full text-search capable engine in order to be searchable with free text. This repo is based on Apache Solr[34], a full-text search engine, which can index JSON objects with the use of Apache Lucene[35]. The ingested textual content will be processed through a pipeline applying stemming, lemmatization and other natural language processing techniques and it is then stored in a tf-idf index. The Apache Solr engine offers a set of features useful for information retrieval such as faceting, hit highlighting, field search and more. The search engine is hosted on an on premise private cloud architecture, following both distribution (sharding) and replication (clustering) techniques for high-availability, redundancy as well as performance and scalability as shown in Figure 68.

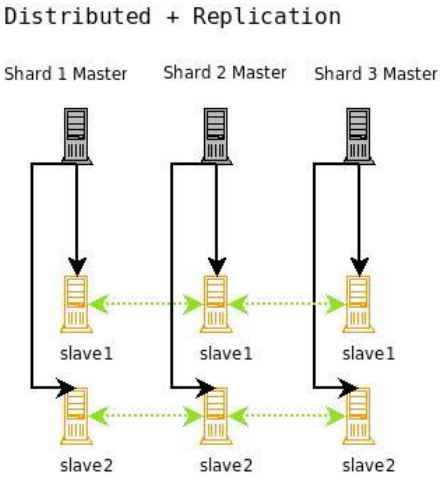

Figure 68: Apache Solr Deployment

Trend Detection

The Trend Detection module is based on the aggregated data, which is ingested automatically inside Truly Media platform and usually monitors a specific topic (e.g., Covid-19 news). As mentioned in the previous section, the content is subsequently analysed through a processing pipeline with Natural Language Processing (NLP) libraries. This involves Named Entity Recognition, (NER), and sentiment extraction. The analysis pipeline is enriching the metadata of each collected document with extra information.

[34] https://solr.apache.org/

[35] https://lucene.apache.org/

The Trend Detection module is implemented as a standalone Java Spring Boot[36] process, which is hosted on a Kubernetes cluster and is running periodically using the CronJob Scheduler[37]. The role of this module is to execute specific queries on the collected data in order to detect peaks or other interesting patterns. The algorithm and the business rules are flexible and customisable by the administrator of the specific monitoring tasks and can also include notifications and alerts. Thus, every time a set threshold is passed, the system will throw an alert and will store this information inside the repository to be visualised when the user is visiting the platform.

Except for the alerting, the service is visualising the data with trend lines showing the occurrence volume of specific entities and hashtags as well as the percentage of negative and positive mentions in a followed topic (red and green colour, respectively). Figure 69 and Figure 70 present some examples of the Trend Detection visualisations based on "COVID-19 vaccination" topic.

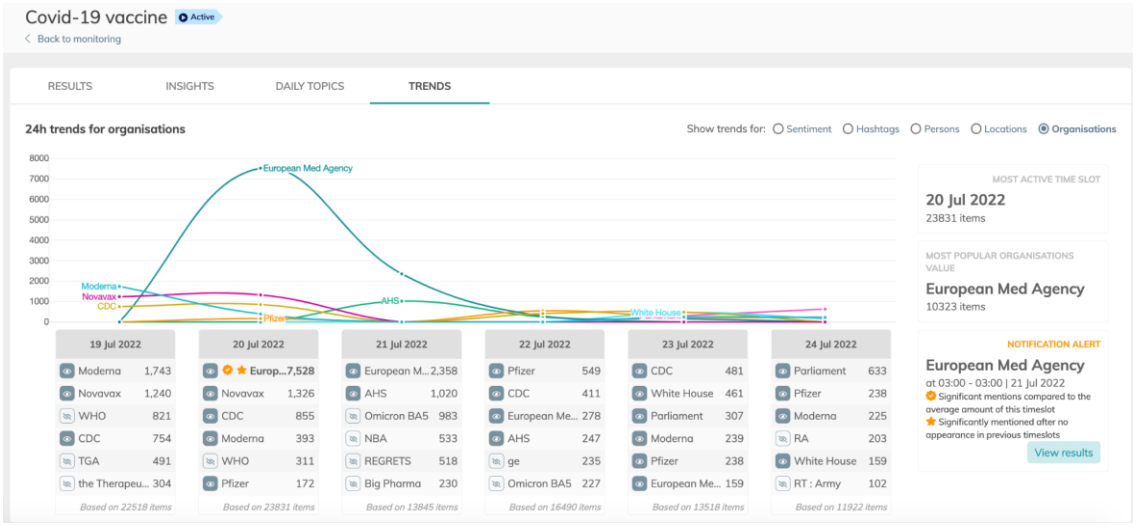

Figure 69: Trend Detection for the topic "COVID-19 Vaccine" during July 2022 (extracted "organization" entities)

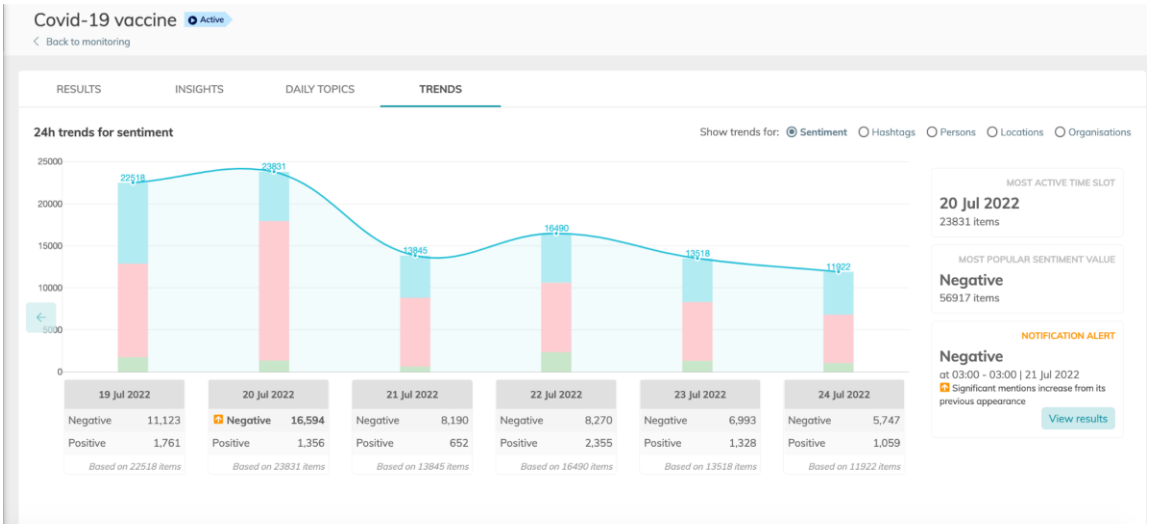

Figure 70: Trend Detection (total/sentiment) for COVID-19 Vaccine during July 2022

[36] https://spring.io/projects/spring-boot

[37] https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

Topic Detection

The Topic Detection feature is using advanced unsupervised AI algorithms in order to create clusters of similar content items (e.g., Tweets, text articles, posts, etc.) and group them together in the form of clusters. It is running on an hourly basis and is updating the clusters on a daily scope. This means that during the 24-hour window of a day, the service runs the clustering algorithm on an hourly interval taking as input the daily tweets collected until that moment. The newly calculated clusters are updating the previously calculated clusters of the same day. This process continues on every hour until the date changes. Then it starts the calculation for the next day with a clean set of topics.

Incoming documents are processed and then summarised and embedded into space using BERT (bi-directional Encoder Representations from Transformers) embeddings and then they are clustered using summary or keywords using the k-means algorithm with auto cluster calculation using the elbow method. The output is a set of the most indicative items of the cluster together with a set of keywords characterising the cluster. The following examples in Figure 71 and Figure 72 show the daily topics as well as the top members of each topic.

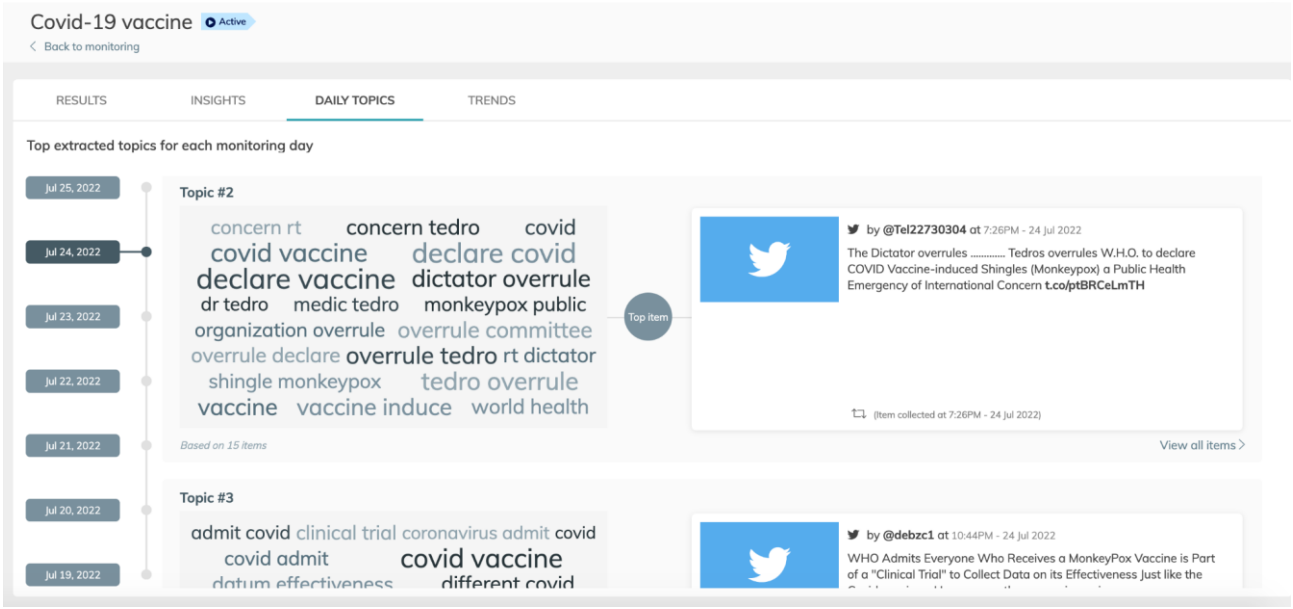

Figure 71: Topic Detection (keyword representation)

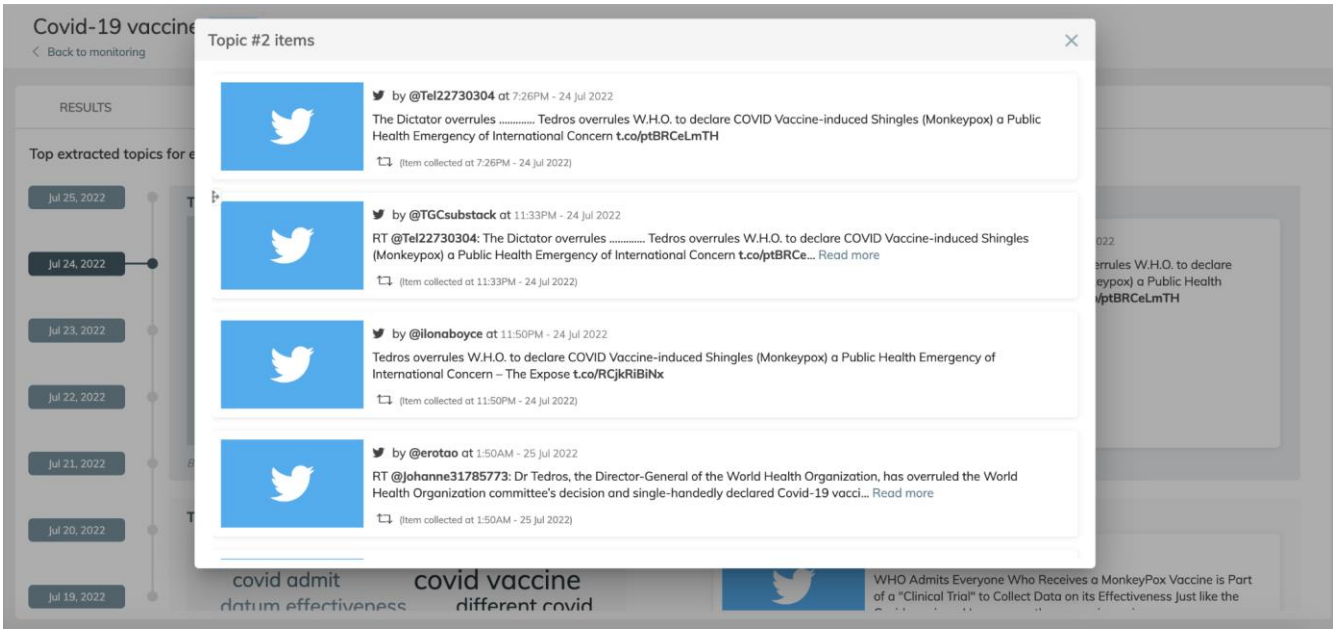

Figure 72: Topic Detection (indicative members of the cluster)

Influencers - Insights

For further analysis of the monitored topic, we have created the "Insights" tab. This dashboard consists of a set of diagrams, visualisations and lists of mentions, which offer an overview over a topic (see Figure 73). The "Insights" view can be customised with the use of facets and filters and it can reveal other important information over the monitored topic, such as the most popular multimedia items, influential profiles, very active profiles over this topic, most popular positive or negative posts, related hashtags, etc. With the use of these visualisations, the end user may get a better understanding of the investigated topic and can retrieve better quality content.

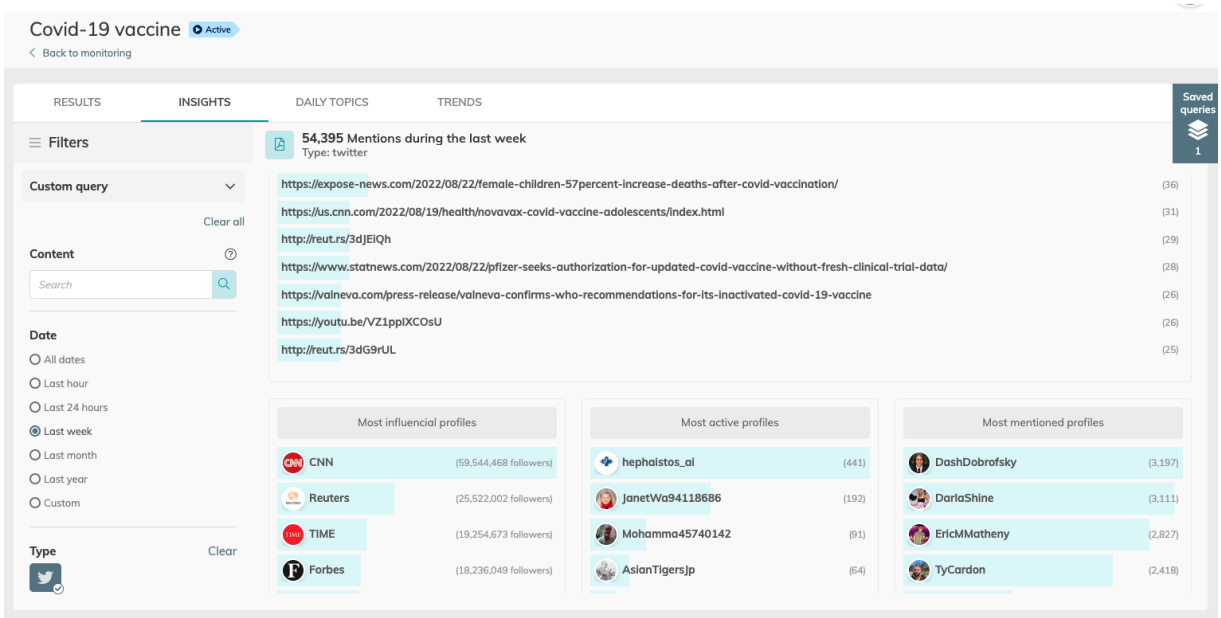

Figure 73: Insights View

7 Conclusions and Future Work

In this deliverable, we have presented the developments in the creation of VR content, as well as enriching and measuring its impact through social media. We have presented Fader and VRodos, which are authoring tools for VR experiences. In these tools, we have achieved unidirectional communication with a MediaVerse node. Namely, the user logins with the same credentials as in the MV node to fetch his or her content such as 360 videos and 3D models, and author VR experiences with them using Fader or VRodos. In year 3, we will refine the developments towards improving the quality of the authoring services, by performing user evaluations and retrieving their feedback. In addition, work will be done towards the direction of providing the advanced experiences back to MediaVerse nodes as videos or links.

As regards social analytics, we will work on the integration of the Truly Media platform with the MediaVerse platform. The Truly Media platform will be available as a third-party service where a MediaVerse user will be able to perform a single sign on by using her/his Social Media account (e.g., Twitter account). We will investigate the possibility of using the Firebase Authentication process (section 6.1) for performing authentication through the Truly Media services. That way the MediaVerse user will have access to the tracking services (section 6.3) and the ability to exchange content between the two platforms. Regarding the content publishing functionality, our plan is to adapt more Social media platforms and more specifically to provide the possibility of publishing content to TikTok, YouTube, and Facebook.

References

Deng, J., Guo, J., Ververas, E., Kotsia, I., & Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5203-5212).

Efron, B., & Tibshirani, R. J. (1994). An introduction to the bootstrap. CRC press.

Guo, J., Deng, J., Lattas, A., & Zafeiriou, S. (2021). Sample and computation redistribution for efficient face detection. arXiv preprint arXiv:2105.04714.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

Li, J., Wang, Y., Wang, C., Tai, Y., Qian, J., Yang, J., ... & Huang, F. (2019). DSFD: dual shot face detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5060-5069).

Liu, Y., & Tang, X. (2020). Bfbox: Searching face-appropriate backbone and feature pyramid network for face detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13568-13577).

Liu, Y., Tang, X., Han, J., Liu, J., Rui, D., & Wu, X. (2020, June). Hambox: Delving into mining high-quality anchors on face detection. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 13043-13051). IEEE.

Liu, Y., Tang, X., Wu, X., Han, J., Liu, J., & Ding, E. (2019). Hambox: Delving into online high-quality anchors mining for detecting outer faces. arXiv preprint arXiv:1912.09231.

Ming, X., Wei, F., Zhang, T., Chen, D., & Wen, F. (2019). Group sampling for scale invariant face detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3446-3456).

Nagendra, S., Baskaran, R., & Abirami, S. (2015). Video-based face recognition and face-tracking using sparse representation based categorization. Procedia Computer Science, 54, 746-755.

Najibi, M., Samangouei, P., Chellappa, R., & Davis, L. S. (2017). Ssh: Single stage headless face detector. In Proceedings of the IEEE international conference on computer vision (pp. 4875-4884).

Pham, M. T., Gao, Y., Hoang, V. D. D., & Cham, T. J. (2010, June). Fast polygonal integration and its application in extending haar-like features to improve object detection. In 2010 IEEE CVPR (pp. 942-949). IEEE.

Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Masia, B., & Wetzstein, G. (2018). Saliency in VR: How do people explore virtual environments?. IEEE transactions on visualization and computer graphics, 24(4), 1633-1642.

Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. Advances in neural information processing systems, 27.

Tang, X., Du, D. K., He, Z., & Liu, J. (2018). Pyramidbox: A context-assisted single shot face detector. In Proceedings of the European conference on computer vision (ECCV) (pp. 797-813).

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International journal of computer vision, 57(2), 137-154.

Yang, B., Yan, J., Lei, Z., & Li, S. Z. (2014, September). Aggregate channel features for multi-view face detection. In IEEE international joint conference on biometrics (pp. 1-8). IEEE.

Yang, S., Luo, P., Loy, C. C., & Tang, X. (2015). From facial parts responses to face detection: A deep learning approach. In Proceedings of the IEEE international conference on computer vision (pp. 3676-3684).

Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). Wider face: A face detection benchmark. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5525-5533).

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., & Li, S. Z. (2017a). S3fd: Single shot scale-invariant face detector. In Proceedings of the IEEE international conference on computer vision (pp. 192-201).

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., & Li, S. Z. (2017b). Faceboxes: A CPU real-time face detector with high accuracy. In 2017 IEEE International Joint Conference on Biometrics (IJCB) (pp. 1-9). IEEE.

Zhu, Q., Yeh, M. C., Cheng, K. T., & Avidan, S. (2006, June). Fast human detection using a cascade of histograms of oriented gradients. In 2006 IEEE CS conf. on computer vision and pattern recognition (CVPR'06) (Vol. 2, pp. 1491-1498). IEEE.

Zhu, Y., Cai, H., Zhang, S., Wang, C., & Xiong, Y. (2020). Tinaface: Strong but simple baseline for face detection. arXiv preprint arXiv:2011.13183.

Annex I: Shader for Chroma Key Removal

In this section, we provide the shader for background Chroma key removal algorithm written with the GLSL language using the JavaScript API. The characteristic of GLSL is that it works on a pixel-wise logic, i.e., the language is applied to all pixels without the need to write for loops, but the GPU handles the application of the algorithm to each pixel. Our custom parameters for the shader are *ThresholdMin, ThresholdMax, red, green, blue*. The vertex shader is the standard GLSL shader for drawing vertices, whereas the fragment shader has the aforementioned parameters for recognizing the background colour and replacing it with transparent pixels. The RGB to YUV matrix is used to transform RGB space to YUV space, which is more efficient for Chroma key removal. In practice, Y (the luminance) is ignored, and only UV colour coding space is used for thresholding. In this manner, the algorithm is luminance independent and depends on the colour coding of each pixel. The algorithm defines three domains using the two thresholds. Under the Min threshold, the Chroma key colour is subtracted from the image, between Min and Max, the Chroma key is subtracted up to a percentage, whereas above the Max threshold is not subtracted at all. Towards the development of the algorithm, the ShaderToy[38] interface for live experimenting with Shader algorithms for JavaScript was used.

```
this.materialIncoming.vertexShader = `
     varying vec2 vUv;

     void main() {
         vec4 worldPosition = modelViewMatrix * vec4( position, 1.0 );
         vec3 vWorldPosition = worldPosition.xyz;
         vUv = uv;
         gl_Position = projectionMatrix * modelViewMatrix * vec4( position, 1.0 );
     }`;
this.materialIncoming.fragmentShader = `
        varying vec2 vUv;
        uniform sampler2D uMap;
        uniform float ThresholdMin;
        uniform float ThresholdMax;
        uniform float red;
        uniform float green;
        uniform float blue;

        mat4 RGBtoYUV = mat4(0.257,  0.439, -0.148, 0.0,
                             0.504, -0.368, -0.291, 0.0,
                             0.098, -0.071,  0.439, 0.0,
                             0.0625, 0.500,  0.500, 1.0 );
       void main() {
             vec2 uv = vUv;
             vec4 tex1 = texture2D(uMap, uv * 1.0);

             //colour to be removed
             vec4 chromaKey = vec4(red / 255.0, green/255.0, blue/255.0, 1);

             //convert from RGB to YUV
             vec4 keyYUV =  RGBtoYUV * chromaKey;

             vec4 yuv = RGBtoYUV * tex1;

             float cc;

             float tmp = sqrt(pow(keyYUV.g - yuv.g, 2.0) + pow(keyYUV.b - yuv.b, 2.0));
             if (tmp < ThresholdMin)
               cc = 0.0;
             else if (tmp < ThresholdMax)
                cc = (tmp - ThresholdMin)/(ThresholdMax - ThresholdMin);
             else
                cc=  1.0;

             gl_FragColor = max(tex1 - (1.0 - cc) * chromaKey, 0.0);
         }`;
```

[38] https://www.shadertoy.com

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.