Human-centred generative AI framework for cultural industries' digital transition

Grant Agreement No 101178362

DELIVERABLE 4.1:

Community Hub: Access, Share, Collaborate.R1

Work Package: 4

LEAD BENEFICIARY:

HSRT

Delivery Date: 19.06.2026

Document Sheet

Project acronym	HAMLET
Project full title	Human-centred generative AI framework for cultural industries' digital transition
Programme	Horizon Europe
Topic	HORIZON-CL2-2024-HERITAGE-01-03
Type of Action	HORIZON Innovation Actions
Grant Agreement	101178362
Start day	1 January 2026
Duration	36 months

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Document Information

Deliverable number	D4.1
Deliverable name	Community Hub Access, Share, Collaborate.R1
Lead beneficiary	HSRT
WP	4
Related task(s)	4.2
Type	R
Reviewers	CHIMERA, CAUSA
Delivery date	31.03.2026
Main author(s)	HSRT, CERTH, TAA
Contributor(s)	CERTH
Dissemination Level	Public (PU)

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Document history

Version	Date	Changes	Reviewer/Contributor
V0.1	2026-04-15	Initial deliverable structure, including chapters for the CCH architecture, frontend implementation and backend implementation.	HSRT
V0.2	2026-04-25	First technical content added, including CCH frontend views, Enablers Catalogue, Projects page, chatroom interface and initial backend/API descriptions.	HSRT
V0.3	2026-05-08	Backend implementation expanded with API surfaces, data model, service-interaction flows, authentication, enabler execution, asset handling and collaboration logic.	HSRT, CERTH
V0.4	2026-05-15	Conclusion, integration activities, next steps and technical references added. Figures, tables and frontend/backend descriptions further consolidated.	HSRT, CERTH, TAA
V0.5	2026-05-24	Internal quality review performed. Editorial corrections, consistency checks, R1/R2 scope clarification, table numbering and terminology alignment applied.	HSRT, TAA
V0.6	2026-05-26	Final pre-review version prepared and sent for internal peer review. Publishable summary, references, abbreviations and document history finalized.	HSRT
V0.7	2026-06-15	Peer-review comments addressed.	CHIMERA, CAUSA, HSRT
V0.8	2026-06-16	Final formatting, quality check and coordinator-level review.	HSRT, CERTH
V0.9	2026-06-16	Final modifications, formatting and coordinator-level review.	HSRT, CERTH
V1.0	2026-06-19	Final version submitted.	HSRT, CERTH

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Publishable summary

This deliverable presents the first release (R1) of the HAMLET Collaborative Community Hub (CCH). The CCH is being established as the central access point to the HAMLET ecosystem. In R1, it enables users to explore the currently included AI-driven enabler interfaces, access project workspaces, interact with selected tools, and collaborate around creative and cultural workflows. The deliverable documents the R1 architecture, frontend interface, selected enabler views, backend API surfaces, data model, and service-interaction logic that form the initial implementation baseline of the Hub.

The R1 release includes the Home Dashboard, Enablers Catalogue, sandbox-style enabler interaction views, project workspace, chatroom functionality, and backend structures for authentication, project management, enabler interaction, asset handling, and collaboration. It also explains how the CCH connects with the wider HAMLET architecture, including the Integration Agent pathway, Data Lakes, Knowledge Graph, Vector Database, and future pilot-facing workflows.

This first release establishes the technical and functional basis for further integration, testing and refinement during the next phases of the project. Future work will focus on controlled consortium and pilot-facing access, additional enabler integration, user testing, improved collaboration features, search and recommendation functions, and continued stabilisation towards a more mature pilot-facing service.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Table of Contents

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

List of Figures

List of Tables

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Abbreviations

Abbreviations	Full name
3D	Three-Dimensional
AI	Artificial Intelligence
API	Application Programming Interface
AUTH	Aristotle University of Thessaloniki
CCH	Collaborative Community Hub
CCI/s	Cultural and Creative Industry/ies
CEDS	Common European Data Space
CERTH	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS
CI/CD	Continuous Integration / Continuous Deployment
CYENS CoE	CYENS Centre of Excellence
ECCCH	European Collaborative Cloud for Cultural Heritage
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable and Reusable
FK	Foreign Key
GANs	Generative Adversarial Networks
GDPR	General Data Protection Regulation
HAMLET	Human-centred generative AI fraMework for culturaL industriEs' digital Transition
HBKsaar	Hochschule der Bildenden Künste Saar
IP	Intellectual Property
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
JSONB	JSON Binary
JWT	JSON Web Token
NPC	Non-Player Character

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

OAuth	Open Authorization
OIDC	OpenID Connect
PDF	Portable Document Format
PK	Primary Key
R1	Release 1
R2	Release 2
RBAC	Role-Based Access Control
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SSO	Single Sign-On
TAA	The Audience Agency
TEXT	Text data type
TIMESTAMP	Timestamp data type
UI	User Interface
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VARCHAR	Variable Character
W3C	World Wide Web Consortium
WP	Work Package
XR	Extended Reality

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

1 Introduction

1.1 Purpose of the deliverable

The purpose of this deliverable is to present the first release (R1) of the HAMLET Collaborative Community Hub (CCH): Access, Share, Collaborate, which constitutes the central access point to the HAMLET ecosystem and its suite of AI-driven enablers. This deliverable reports the design, development, and R1 implementation baseline of the CCH, providing an overview of its architecture, functionalities, and role within the broader HAMLET framework. It focuses on the current frontend structure, selected enabler-related views, backend foundations, API surfaces, data model, and integration-ready architecture. It also clarifies which functions are already available at this stage and which will be further developed toward the 2nd release. At its core, the CCH is conceived as a unified digital platform for accessing, exploring and using HAMLET enablers. In R1, this is implemented through the current frontend structure, the enabler interfaces currently included in the Hub, and backend/API foundations described in this document. It acts as a frontend interface supported by backend services and API structures, bridging user interaction with the AI service components and preparing coherent workflows across multiple tools and modalities. More specifically, this deliverable aims to:

- Present the **overall R1 architecture of the CCH**, including its frontend components (user interface, interaction flows, visualization modules) and backend infrastructure (API gateway, service orchestration, Integration Agent pathway, and data handling mechanisms).
- Describe the **core R1 functionalities** of the CCH, including access to selected AI enablers, basic Hub/enabler interaction functionalities, asset-management foundations, and collaborative features such as project spaces, sharing, reuse, and co-creation concepts.
- Detail the **API structures** and **integration mechanisms** that allow the CCH to communicate with distributed enablers via standardized APIs, supporting interoperability, scalability, and modular extensibility.
- Introduce the **agent-based interaction paradigm**, where users can interact with the platform through natural language queries. This interaction model is supported by the Integration Agent, which progressively enables the selection and composition of appropriate enablers to fulfil user requests.
- Present a set of interface examples and **representative use case scenarios**, illustrating how different user groups (e.g., artists, cultural institutions, developers) can leverage the CCH to perform tasks such as content generation, narrative design, immersive experience creation, and audience-driven content adaptation.
- Describe the **data flow and interaction pipelines**, highlighting how user inputs, project information, assets and service requests are structured in R1 and prepared for broader multi-enabler workflows.
- Provide an initial **specification of the technical implementation**, including frontend and backend components, API endpoints, system requirements, and deployment considerations.
- Establish the **baseline version (R1)** of the CCH, which will serve as the foundation for future enhancements, optimization, pilot-facing testing, and integration activities towards the subsequent release (R2).

This deliverable establishes the R1 baseline for the HAMLET CCH as a human-centred, AI-powered collaborative environment, supporting access to AI technologies while fostering innovation, creativity, and knowledge exchange within the CCIs.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

1.2 Relationship with other deliverables

This deliverable is connected to several HAMLET deliverables, as it consolidates prior design and development efforts while enabling subsequent integration, validation, and exploitation activities. The CCH R1 serves as the operational entry point to the HAMLET ecosystem and functions as a convergence layer that bridges architectural definition (WP1), enabler development (WP2 and WP3), system integration (WP4), pilot validation (WP5), ethics, community-building and policy activities (WP6), and dissemination and exploitation activities (WP7).

At the architectural level, this document builds upon the outputs of WP1, which define the project's requirements, use cases, and the overall Human-Centered AI framework [1],[2]. The conceptual architecture and design principles established in WP1 are here translated into a concrete implementation, including the frontend interface and the backend orchestration mechanisms. In this sense, the CCH translates the high-level architecture into an R1 implementation baseline that supports user interaction, selected Hub/enabler functionalities, and future service integration.

With respect to WP2, this deliverable provides the access and orchestration layer for the horizontal AI enablers, such as procedural content generation, cultural content adaptation, audience engagement analytics, and the Integration Agent. While WP2 deliverables define the individual capabilities and technical specifications of these enablers, the present deliverable provides the first Hub-level structures for exposing selected functionality through standardized APIs and preparing their broader combined use through a single interface. As such, it represents a first step toward making WP2 enablers accessible and interoperable services within a broader ecosystem.

Similarly, the deliverable is closely linked to WP3, which focuses on industry specific enablers, including music-to-movement translation, immersive design integration, adaptive narrative generation, and dance analysis and generation. The CCH provides an initial common interaction layer for these tools, preparing their integration into broader creative workflows. Through this integration pathway, the CCH is expected to facilitate cross-modal pipelines, where outputs from WP2 enablers can be seamlessly combined with WP3 applications to support complex creative processes, such as generating immersive experiences from textual prompts or combining audio-driven motion synthesis with virtual environments.

Within WP4, this deliverable establishes the initial system integration by connecting the enablers through a unified architecture, defining the API communication layer, and introducing the agent-based orchestration mechanism. It also provides the baseline implementation upon which subsequent WP4 deliverables will build, focusing on system optimization, extended integration, and adaptation to pilot-specific requirements.

The relationship with WP5 is important, as the CCH serves as the primary interface for pilot execution and user interaction. The use cases and functionalities described in this deliverable are aligned with the set pilot scenarios and will be validated through real-world deployments, user feedback, and performance evaluation. In this context, the current version of this document establishes the initial platform capabilities, while WP5 activities will guide further refinement and enhancement in subsequent iterations.

All in all, Table 1 summarises the main CCH requirements and specifications that were most relevant for D4.1. This table indicates how the latter are addressed in the R1 release, and identifies follow-up implementation needs for the R2 release.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Table 1: Alignment with prior CCH requirements and next development priorities.

Requirement area	Source	Current focus	Next development need
Hub access point	D1.2, D1.3	Dashboard, navigation, Catalogue, Projects	Extend pilot-specific journeys
Enabler catalogue and access	D1.2, D1.3	Selected enabler views and API descriptions	Complete metadata and workflows
Hub/enabler interaction	D1.3	Basic selected interactions and API structures	Complete end-to-end enabler workflows
Project collaboration	D1.2, D1.3	Projects Workspace and chatroom basis	Add roles, templates and collaboration flows
Resource sharing	D1.3, D9.5	Asset/data model and backend basis	Implement Resources page and upload/share/reuse flows
Matchmaking and competency exchange	D1.2, D1.3	Navigation/screen basis where available	Add profiles, skills and matching logic
IPR, licensing and provenance	D1.2, D9.5, WP7	Asset/project structures; Contract Editor if included	Add licence, attribution, provenance and agreement fields
Ethics and responsible use	D1.3, D9.1, D6.2	Not yet fully evidenced in frontend	Add Ethics section and responsible-use guidance
Integration Agent pathway	D1.3	Hub Assistant concept and API structures	Complete orchestration and intent mapping
Human control and transparency	D1.2, D9.1	Enabler views and descriptors	Add review/edit/approve states and output explanations
Validation and adoption	D1.2, D6.2, D9.4	R1 technical baseline	Add TAA/user feedback, analytics and usability testing

1.3 R1 status and pilot handover relevance

The R1 release documented in this deliverable represents the first implementation baseline of the HAMLET CCH. It brings together the core platform elements required for initial consortium use, including the Home

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Dashboard, Enablers Catalogue, enabler interfaces currently included in R1, Projects page, chatroom functionality, backend API surfaces, data model, authentication structures, asset-handling logic and service-interaction foundations. As such R1 provides the operational basis for controlled consortium access and for the next phase of pilot-facing testing, refinement and integration towards R2.

Controlled consortium access to the CCH environment will focus on the R1 components already implemented and described in this deliverable. The first access route will cover the platform areas that are available in the current implementation, while R2 will extend this baseline with additional enabler connections, more complete end-to-end workflows, pilot-specific adaptations and broader user-facing refinements. The enabler interfaces currently included in R1 reflect the present integration scope of the CCH and provide the first access patterns for moving from enabler discovery to practical use inside the Hub.

At this stage, validation covers internal technical consolidation and consortium-level review of the implemented baseline. Broader user and pilot validation will follow through controlled consortium access, pilot feedback loops and subsequent WP5 activities. In this way, R1 acts as the handover point between platform implementation and pilot-facing validation, while R2 will consolidate the remaining CCH functionality and pilot-specific refinements. Table 2 summarizes the status of the main CCH areas at R1 stage, while Table 3 summarizes the current CCH coverage of HAMLET enablers.

Table 2: Status of the main CCH areas at R1 stage

CCH area	R1 status	Initial check	R2 follow-up
Home Dashboard	Implemented	Access, navigation and entry points	Improve onboarding and pilot-specific journeys
Enablers Catalogue	Implemented	Enabler discovery and catalogue structure	Add metadata, filtering & broader enabler coverage
Enabler interfaces currently included in R1	Implemented	Inputs, outputs and user flow	Add remaining enabler interfaces and complete workflows
Projects page	Implemented	Project structure, members, assets and activity	Add templates, export options and pilot-specific flows
Chatroom	Implemented	Basic communication and collaboration	Improve notifications and links with project workflows
Backend APIs	Implemented	Users, projects, assets, enablers and service interactions	Stabilise deployment and external integrations

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Data model and asset handling	Implemented	Project, asset, user and enabler-instance logic	Add retention, cleanup, provenance and governance refinements
Integration Agent pathway	Implemented at R1 integration level	Routing and orchestration logic	Extend orchestration, intent mapping and multi-enabler execution
Resource sharing	Implemented through project & asset structures	Asset association with project workspaces	Add fuller upload, sharing & reuse flows
Matchmaking and competency exchange	Planned for R2	—	Add profiles, skills & matching logic
IPR, licensing and provenance	Basic asset/project structure in place	Association of assets, projects, users and outputs	Add licence, attribution and provenance fields
Ethics and responsible-use guidance	Planned for R2	—	Add responsible-use guidance & governance links
Human control and transparency	Partially supported	User interaction with generated outputs	Add review, edit, approve and explanation states
Controlled consortium access	Under preparation	R1 platform and current enabler interfaces	Provide consortium/pilot access after stability, security, legal & accessibility checks
User/pilot validation	Planned for the next phase	Internal technical review	Conduct controlled-access testing, WP5 pilot feedback & R2 refinement

Table 3: The current CCH coverage of HAMLET enablers

Enabler / tool	R1 coverage	R2 follow-up
Adaptive Narrative Enabler	Included in the R1 CCH interface	Confirm pilot execution pathway and integration depth
Procedural Content Generator / 3D Asset Synthesis	Included in the R1 CCH interface	Improve export/download and game-engine workflow support
Cultural Content Adaptation Enabler	Included in the R1 CCH interface	Complete pilot-specific testing and technical refinements
Enabler Integration Agent	Included in the backend integration logic	Extend orchestration and multi-enabler workflows

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Audience Engagement Analytics Enabler	Registered through the CCH catalogue/API logic; dedicated interface to follow	Add pilot-relevant analytics workflows in R2
Music-to-Movement Translation Enabler	To be exposed through the CCH in the next integration phase	Add dance/performance workflows in R2
Immersive Design Integration Enabler	To be exposed through the CCH in the next integration phase	Add immersive-experience workflows in R2
Dance Style Analysis and Generation Enabler	To be exposed through the CCH in the next integration phase	Add dance-analysis/generation workflows in R2
Combined / multi-enabler workflows	Supported by the R1 orchestration basis	Consolidate cross-enabler workflows in R2

2 HAMLET Architecture

The architecture of the HAMLET platform was defined and expanded in WP1 and documented in detail in D1.3. This chapter briefly recalls the architectural elements most relevant to the R1 implementation of the CCH.

2.1 High Level Viewpoint

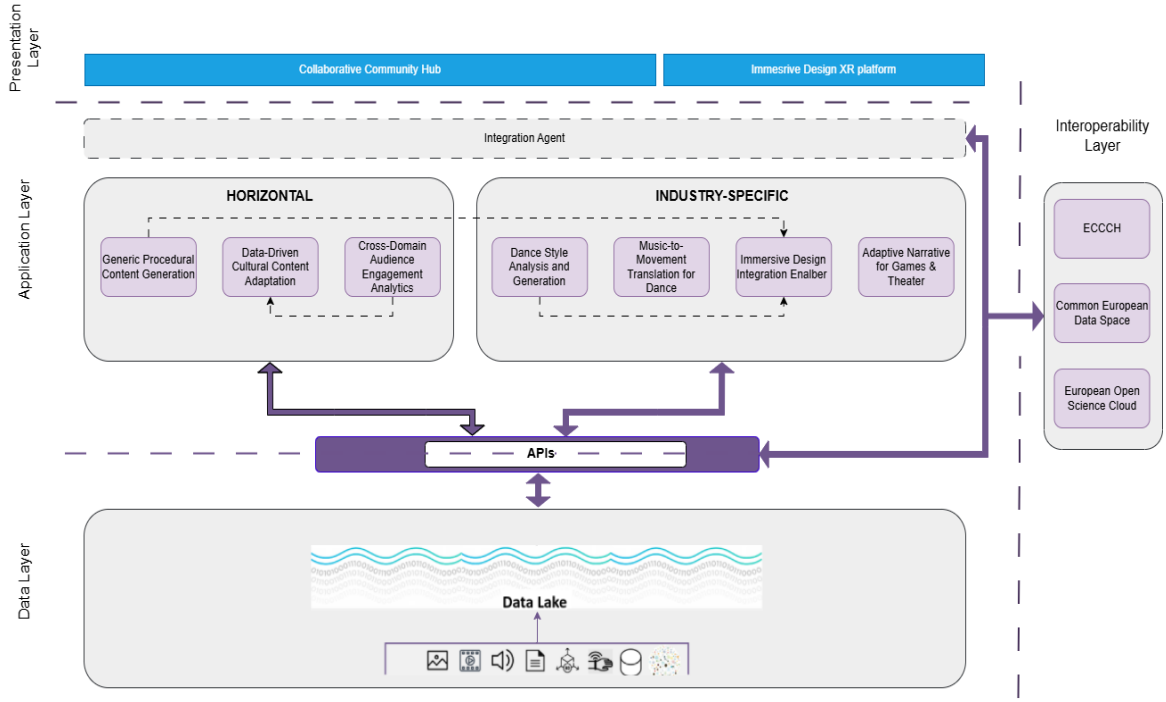

Figure 1 HAMLET Architecture High Level viewpoint

The HAMLET architecture is structured into four interconnected layers that collectively support user interaction, AI-driven processing, data management, and interoperability with external infrastructures. This layered design ensures a clear separation of concerns while enabling seamless data flow and coordination across

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

components. At a high level, the system is designed to translate user intent into multimodal workflows by coordinating AI enablers, data infrastructure and standardized interfaces.

The Presentation Layer constitutes the primary interaction point between users and the system, providing intuitive and accessible interfaces tailored to the needs of Cultural and Creative Industries. It includes the CCH, which provides the user-facing environment for communication, co-creation and access to AI tools, as well as the Immersive Design XR Platform, which supports extended reality experiences for artistic experimentation and hybrid performances. An integrated interactive assistant enhances usability by guiding users through the platform and facilitating the discovery and use of available functionalities.

The Application Layer forms the operational core of the system, supporting AI-driven processes and communication between components. It brings together horizontal enablers that provide general-purpose AI capabilities across domains and industry-specific enablers that tailor these capabilities to particular creative sectors. Central to this layer is the Integration Agent, which supports natural-language interaction and provides the pathway for selecting and combining appropriate enablers. This design prepares the system to support more complex multimodal pipelines while maintaining modularity and flexibility.

The Data Layer underpins the entire architecture by handling the storage, organization, and retrieval of multimodal data, including images, audio, video, 3D assets, and metadata. Leveraging data lake technologies, it ensures scalable storage, efficient access, and secure management of information. This layer supports both real-time processing and long-term knowledge representation, acting as the backbone for all AI operations and enabling consistent data availability across the system. Finally, the Interoperability Layer ensures that HAMLET is seamlessly connected with broader European digital infrastructures, such as the European Collaborative Cloud for Cultural Heritage (ECCCH), the Common European Data Space, and the European Open Science Cloud (EOSC). Through standardized APIs and data exchange mechanisms, this layer enables cross-platform collaboration, supports FAIR and open data principles, and facilitates the integration of external datasets and services into the HAMLET ecosystem. Key architectural characteristics include:

- A layered and modular design that supports scalability and extensibility.
- An agent-based interaction pathway enabling intuitive, natural-language interaction.
- Multimodal integration logic supporting workflows across text, audio, motion and 3D content.
- Strong emphasis on interoperability and open-data alignment within the European digital ecosystem.

The functional viewpoint constitutes a core perspective of the HAMLET architecture, following the 4+1 model, and focuses on how the system operates and how its components interact to deliver functionality. It represents the first level of decomposition of the high-level architecture, providing a structured view of the enablers and modules and their roles within the framework.

This viewpoint illustrates the functional relationships and interaction flows between components, emphasizing how individual enablers collaborate to realize the system's capabilities. Each module is presented in terms of its core functionality, while the connections between them highlight the exchange of data and the orchestration of processes across the platform.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

2.2 Functional Viewpoint

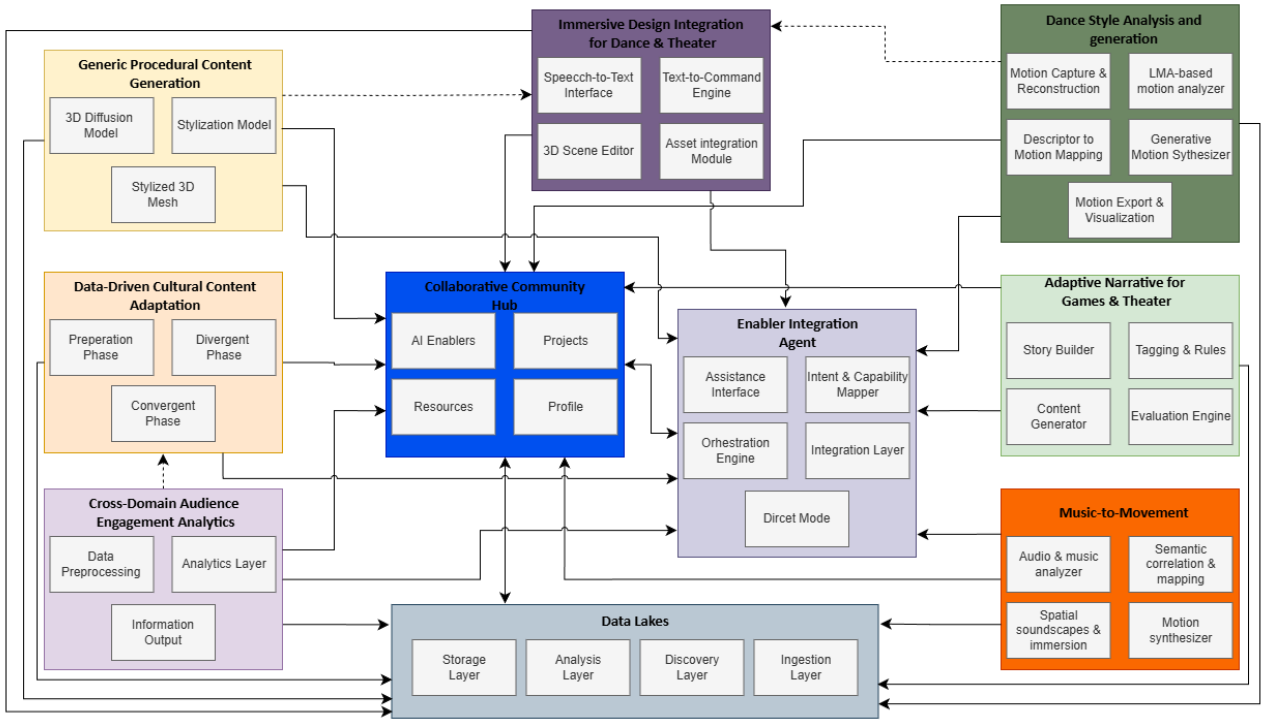

Figure 2 Functional viewpoint of HAMLET architecture

As depicted in Figure 2, the functional viewpoint maps the system into distinct components aligned with project tasks, with interaction pathways demonstrating how user requests are processed and propagated through the HAMLET framework. This perspective enables a clear understanding of the system's operational logic, supporting both integration and further development of the platform.

2.3 Authentication and Authorization

The following subsection summarises the authentication and authorization approach foreseen for the HAMLET platform and reflected in the R1 backend/API design. The HAMLET is designed around a centralized authentication and authorization framework to ensure secure, seamless, and consistent access across its distributed components, including the CCH, the Immersive Design XR Platform, and the suite of AI-driven enablers developed in WP2 and WP3. This framework enables users, such as cultural and creative professionals, artists, researchers, developers, and public participants, to interact with multiple services through a unified and secure identity management system.

At the core of this framework lies a Single Sign-On (SSO) mechanism, which allows users to authenticate once and gain access to all connected services within the HAMLET ecosystem. Each system component, including frontend applications and backend enablers, is registered as a trusted client within the identity management infrastructure, ensuring consistent authentication flows and secure communication based on widely adopted standards such as OAuth 2.0 [4] and OpenID Connect (OIDC).

Upon successful authentication via the CCH or any integrated interface, users are issued a JSON Web Token (JWT) [5] that encapsulates identity attributes, assigned roles, and access permissions. These tokens are propagated across the platform and used to authorize API requests to the various enablers and services. This approach ensures that each component can independently verify user permissions without duplicating authentication logic, thus supporting a scalable microservices architecture. The platform enforces Role-Based

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Access Control (RBAC) policies to regulate access to functionalities and resources. Different user roles, such as administrators, content creators, developers, and general users, are assigned distinct privileges, enabling fine-grained control over actions such as accessing AI enablers, managing generated assets, using workflow-related functions, or interacting with collaborative features within the CCH. This ensures both security and flexibility in supporting diverse user groups and use cases. From an architectural perspective, authentication and authorization are tightly integrated with the Integration Agent and API layer, ensuring that all orchestrated workflows and cross-enabler interactions comply with access policies. This is particularly important in scenarios involving multi-step pipelines, where user permissions must be validated across multiple services dynamically.

Furthermore, the HAMLET platform is designed to support federated identity integration, allowing external organizations and institutions to authenticate users through their own identity providers (e.g., SAML, institutional OAuth systems). This capability enhances interoperability, facilitates user onboarding, and promotes trust across collaborative environments. Overall, the centralized authentication and authorization framework enables HAMLET to maintain a secure, interoperable, and user-friendly ecosystem, ensuring compliance with European data protection regulations while supporting scalable and flexible access to advanced AI-driven services.

2.4 HAMLET Data Lakes

The HAMLET platform is expected to generate and manage a wide range of heterogeneous data arising from user interactions, AI-driven processes, and multimodal content generation workflows. These data span multiple modalities, including text, images, audio, video, motion data, 3D assets, and semantic metadata, requiring a scalable and flexible infrastructure capable of handling both structured and unstructured information. To address this complexity, HAMLET adopts a Data Lake-based architecture, supported by specialized storage and retrieval components that collectively form the platform's knowledge infrastructure. At the core of this architecture lies the multimodal Data Lake, which is designed to serve as the primary repository for raw and processed data. It is designed to efficiently store large-scale datasets generated across the platform, including AI-generated assets (e.g., 3D models, motion sequences, XR scenes), user-provided inputs, and intermediate processing outputs. The Data Lake supports distributed storage and ensures high availability, scalability, and efficient access across all system components.

For R1, the use of the Data Lake does not imply that all generated content, intermediate outputs or obsolete assets will be retained permanently. The CCH follows an asset-registration approach in which generated content can be stored or referenced through project-level asset records, while detailed retention and cleanup rules remain subject to further refinement. Local-only storage may still be relevant for specific temporary or development-side processes, but it is not the general access and management model of the R1 CCH. To address the risk of large volumes of generated data, the R2 and pilot-facing integration work will refine data-retention rules, project-level ownership responsibilities, storage quotas, automated cleanup mechanisms and compression options for saved files, in alignment with Data Lake governance, access-control requirements and the practical needs of each pilot workflow.

Complementing the Data Lake, the platform incorporates a Knowledge Graph, which manages structured metadata and semantic relationships between entities. This includes information about generated assets, user interactions, contextual annotations, and cross-modal links between different data types. The Knowledge Graph enables advanced querying and reasoning capabilities, supporting interoperability and enhancing the interpretability of AI outputs within the HAMLET ecosystem.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

To support AI-driven functionalities, HAMLET also integrates a Vector Database, which stores high-dimensional embeddings derived from multimodal data (e.g., text, images, audio, and motion). These embeddings enable efficient similarity search, cross-modal retrieval, and semantic matching, which are essential for functionalities such as content recommendation, asset reuse, and adaptive content generation. In addition, a Metadata and Asset Management Layer is employed to track the lifecycle of digital assets across the platform. This includes information related to creation processes, provenance, usage context, and relationships between original and derived content, supporting transparency and reproducibility in AI-assisted workflows.

Access to all data services is mediated through an API Gateway, which provides a unified interface for secure and standardized communication between the Data Layer, the Application Layer, and external systems. The API Gateway ensures consistent data access, enforces authentication and authorization policies, and facilitates interoperability with external infrastructures such as the European Open Science Cloud (EOSC) and the Common European Data Space. The HAMLET Data Lake and knowledge infrastructure are designed to enable the efficient storage, organization, and retrieval of multimodal data, forming the backbone of the platform's AI capabilities. This design supports scalable data processing, cross-modal integration, and seamless interaction between enablers, while ensuring compliance with data governance, interoperability, and FAIR data principles.

3 Collaborative Community Hub frontend implementation (UI)

3.1 Home Dashboard Page

Figure 4 presents the main interface of the HAMLET CCH which serves as the primary access point for users to interact with the platform's enablers, projects, and collaborative functionalities. The interface is designed following a user-centric and modular layout, enabling intuitive navigation, seamless interaction, and efficient access to system capabilities.

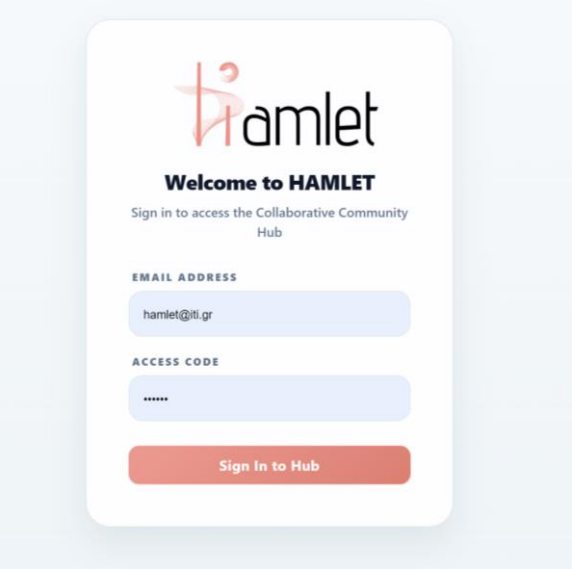

Figure 3: R1 CCH access screen and Home Dashboard.

The R1 interface includes an access screen for signing into the CCH before entering the main Home Dashboard (Figure 3). This supports the authentication and user-session logic described in Section 2.3 & the backend/API struc-tures described in Section 4.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

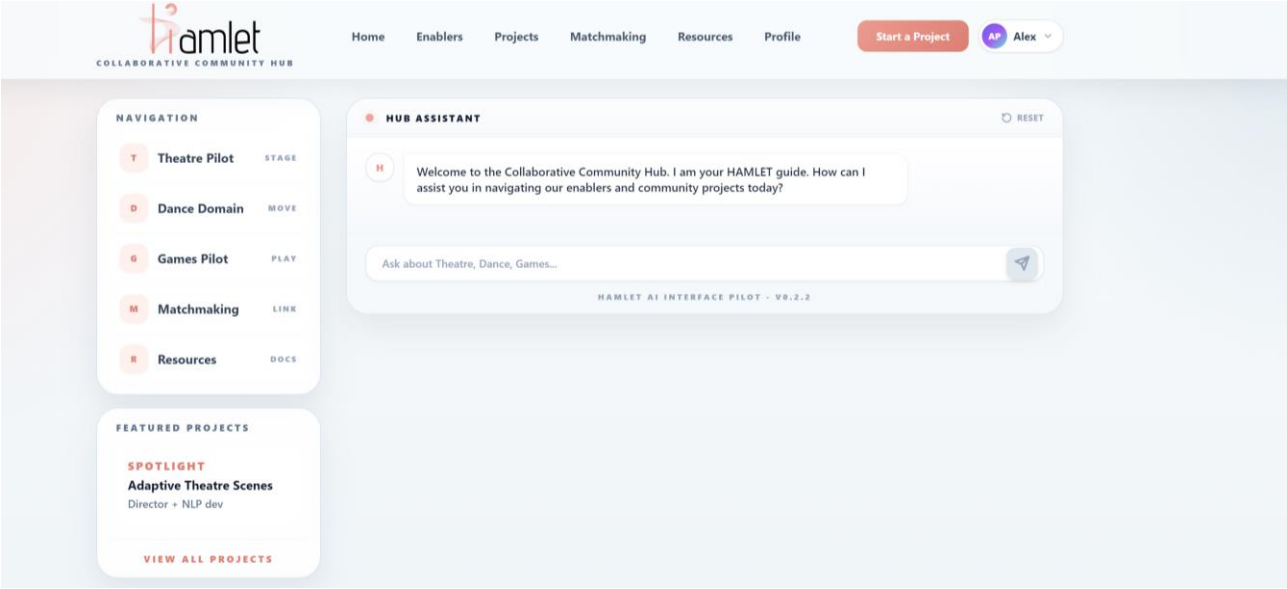

Figure 4 HAMLET Home CCH page

At the top of the interface, a global navigation bar provides direct access to the core platform sections, including Home, Enablers, Projects, Matchmaking, Resources, and User Profile. Additionally, a prominent "Start a Project" action button allows users to initiate new creative workflows, while user account controls are accessible on the right-hand side, supporting personalization and session management.

On the left-hand side, a **navigation panel** organizes the platform's main operational domains and pilot areas. These include:

- Theatre Pilot
- Dance Pilot
- Games Pilot(s)
- Matchmaking
- Resources

In R1, this structure provides access to the main domain-specific areas and prepares the basis for further pilot-specific functionality. The central component of the interface is the Hub Assistant, which acts as an interactive entry point to the system. In the R1 implementation, this assistant is connected to the Integration Agent pathway and supports natural language interaction, allowing users to query the system, discover enablers, and navigate available services. The assistant provides contextual guidance and suggestions, lowering the barrier to accessing AI-driven workflows through the R1 interface. A chat input field allows users to submit queries, while system responses are displayed in a conversational format, enhancing usability and engagement.

Below the navigation panel, the "Featured Projects" section presents project examples and provides the basis for exploring or joining project workspaces as the Hub moves towards controlled consortium use. This section supports the collaborative and co-creative nature of the platform by promoting visibility and knowledge sharing across users. Overall, the interface integrates navigation, interaction, and collaboration functionalities into a cohesive environment. It combines:

- a structured navigation system for domain access,
- an AI-driven assistant for intelligent interaction, and

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

- project-based collaboration features for community engagement.

This design reflects the R1 implementation objective of the HAMLET CCH, which is to establish a centralized, intuitive and AI-assisted access layer for collaboration and enabler interaction.

3.2 Enablers Catalogue

Figure 5 presents the Enablers Catalogue interface of the HAMLET CCH, which provides users with a structured overview of the AI-driven tools and services available through the R1 platform. In R1, this interface supports discovery and initial use of the enabler interfaces currently included in the Hub.

The page is organized around a card-based layout, where each enabler is represented as an individual component containing key information, including its name, associated domain (e.g., Theatre, Dance, Games, Analytics), and descriptive tags that highlight its core functionality (e.g., XR, motion, diffusion, semantics). This design allows users to quickly understand the purpose and technical focus of each enabler at a glance. Each enabler card includes two primary interaction options:

- "Add to Project", which allows users to incorporate the selected enabler into an ongoing or newly created project workflow, supporting modular composition of AI pipelines.
- "Launch Sandbox", which provides access to the R1 enabler-specific interaction environment where users can experiment with the available R1 functionality without affecting project-level configurations.

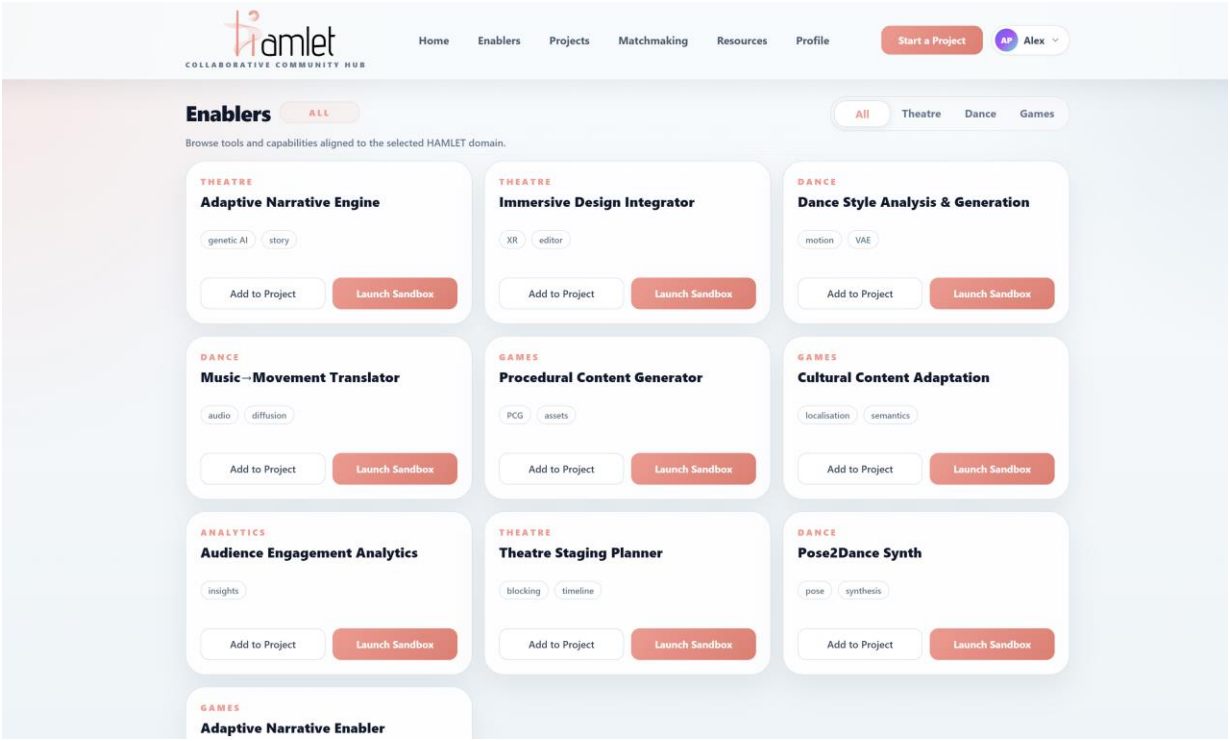

Figure 5 Enablers Catalogue Page

The catalogue brings together both horizontal enablers (e.g., procedural content generation, cultural content adaptation, audience analytics) and industry-specific enablers (e.g., music-to-movement translation, immersive design, dance synthesis), reflecting the layered architecture of HAMLET. This unified presentation enables seamless navigation across general-purpose and domain-specific capabilities.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Overall, the Enablers Catalogue serves as a central discovery and interaction layer, bridging users with the underlying AI services. It supports:

- modular workflow construction,
- rapid prototyping through sandbox environments, and
- efficient exploration of multimodal capabilities across creative domains.

This interface plays a key role in operationalizing the HAMLET ecosystem, making advanced AI functionalities accessible, composable, and directly usable within collaborative creative processes.

Figure 6 presents an example of an enabler selected from the HAMLET Enablers Catalogue, specifically the Adaptive Narrative Enabler, showcased within its interactive sandbox environment. This view illustrates how individual enablers can be accessed, configured, and executed through the R1 CCH, providing users with hands-on interaction and real-time content generation capabilities. The interface follows a structured, editor-based layout, enabling users to define and manage narrative elements in a modular and intuitive manner. On the left-hand side, a hierarchical navigation panel organizes narrative components such as locations, items, non-player characters (NPCs), and tags. This allows users to construct complex narrative environments by systematically defining their constituent elements. The central panel provides detailed configuration options for the selected entity (e.g., a location such as “Harbor”), including:

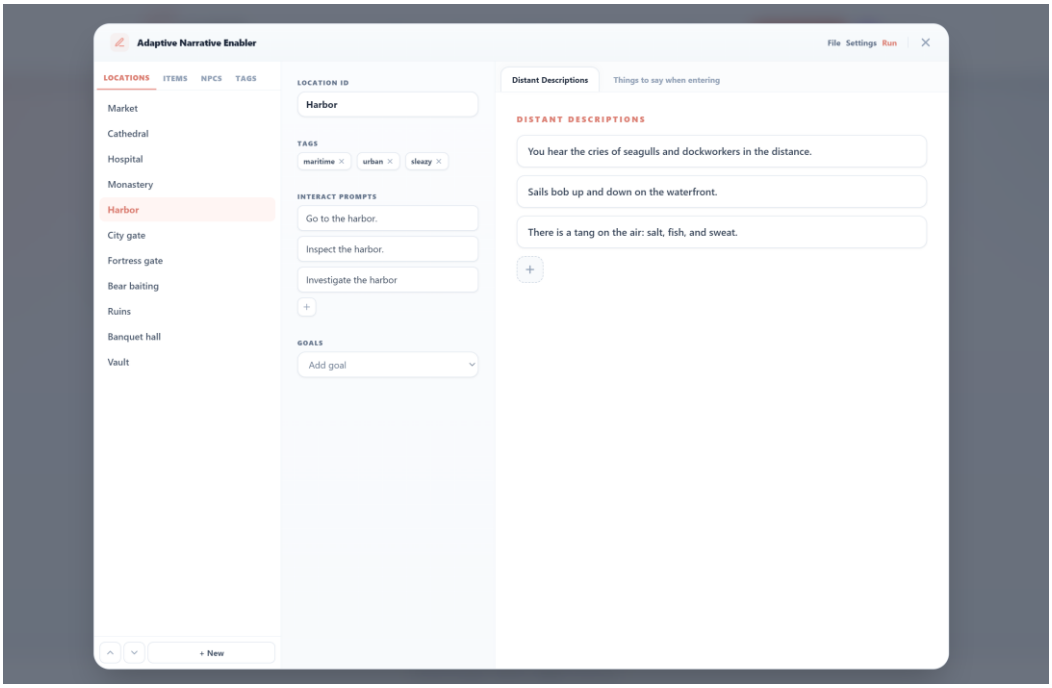

Figure 6 Adaptive Narrative Enabler page

- Metadata definition through tags (e.g., maritime, urban, sleazy), enabling semantic characterization of narrative elements,
- Interaction prompts, which define possible user actions or narrative triggers, and
- Goals, supporting the structuring of narrative progression and objectives.

On the right-hand side, the interface presents dynamically generated narrative content, such as contextual descriptions and storytelling elements (e.g., environmental descriptions or sensory cues). These outputs are generated through the enabler’s AI-supported functionality and can be iteratively refined by the user, supporting a co-creative workflow between human input and AI generation.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Additional controls (e.g., Run, Settings, File) allow users to execute the narrative generation process, manage configurations, and refine outputs, further emphasizing the interactive nature of the sandbox environment. This example demonstrates how enablers within the HAMLET catalogue are discoverable and usable through the R1 CCH interface, enabling users to:

- experiment with AI-driven functionalities in isolation,
- iteratively develop creative content, and
- use generated outputs within broader project workflows

Overall, the Adaptive Narrative Enabler exemplifies the transition from enabler discovery (catalogue view) to initial enabler use through the CCH, highlighting the practical value of the HAMLET platform in supporting structured, AI-assisted creative processes.

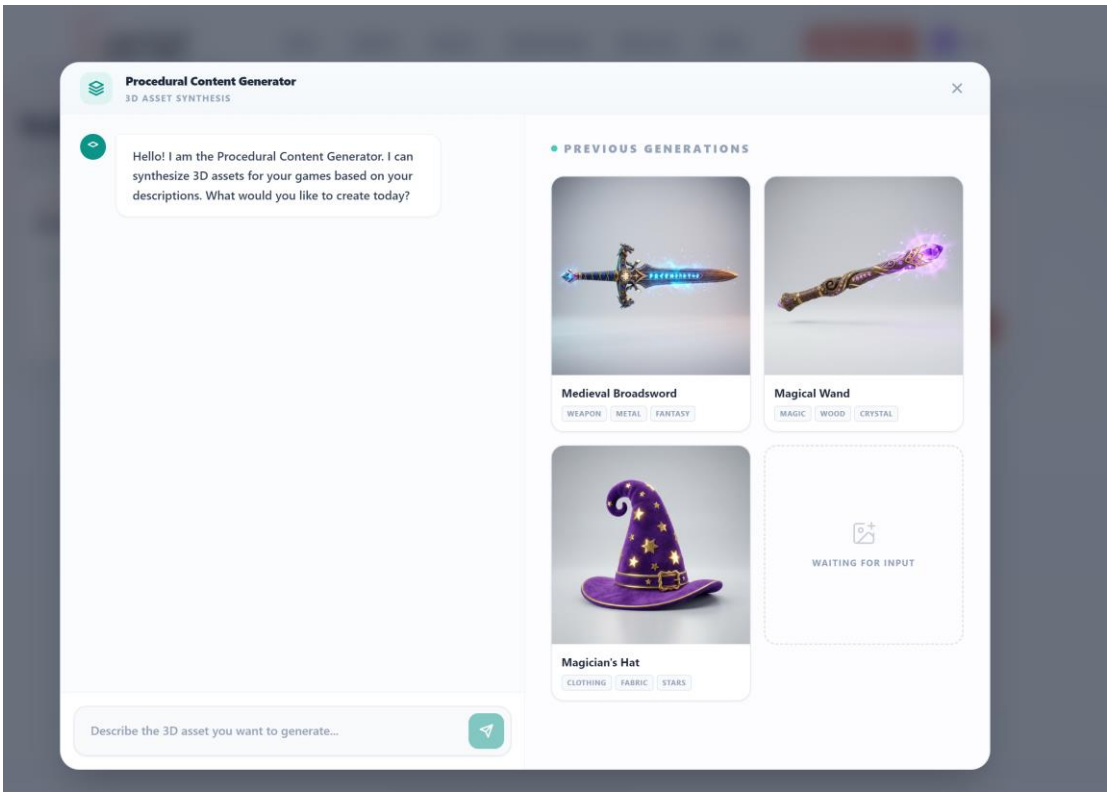

Figure 7: Procedural Content Generator/3D Asset Synthesis R1 interaction view.

Figure 7 presents the Procedural Content Generator/3D Asset Synthesis interaction view, which extends the Enablers Catalogue examples beyond narrative-oriented tools. The interface allows users to describe the type of 3D asset they wish to generate through a prompt-based input field and to review generated outputs in a visual gallery. In the example shown, assets such as a sword, wand and magician's hat are presented as generated or previously generated content items.

This view demonstrates how the CCH supports interaction with multimodal creative enablers through the R1 sandbox-style environment. While the Adaptive Narrative Enabler focuses on text and narrative workflows, the Procedural Content Generator illustrates how the same catalogue-to-interaction pattern can be applied to visual and 3D asset creation. The Procedural Content Generator example shows how users can move from enabler discovery to hands-on creative production within the R1 implementation.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Figure 8 illustrates the Cultural Content Adaptation Enabler in its R1 sandbox environment, as accessed through the HAMLET Enablers Catalogue. This interface demonstrates how users can interact with the enabler through a conversational, AI-assisted workflow to generate and adapt narrative content in a culturally aware manner within the R1 implementation.

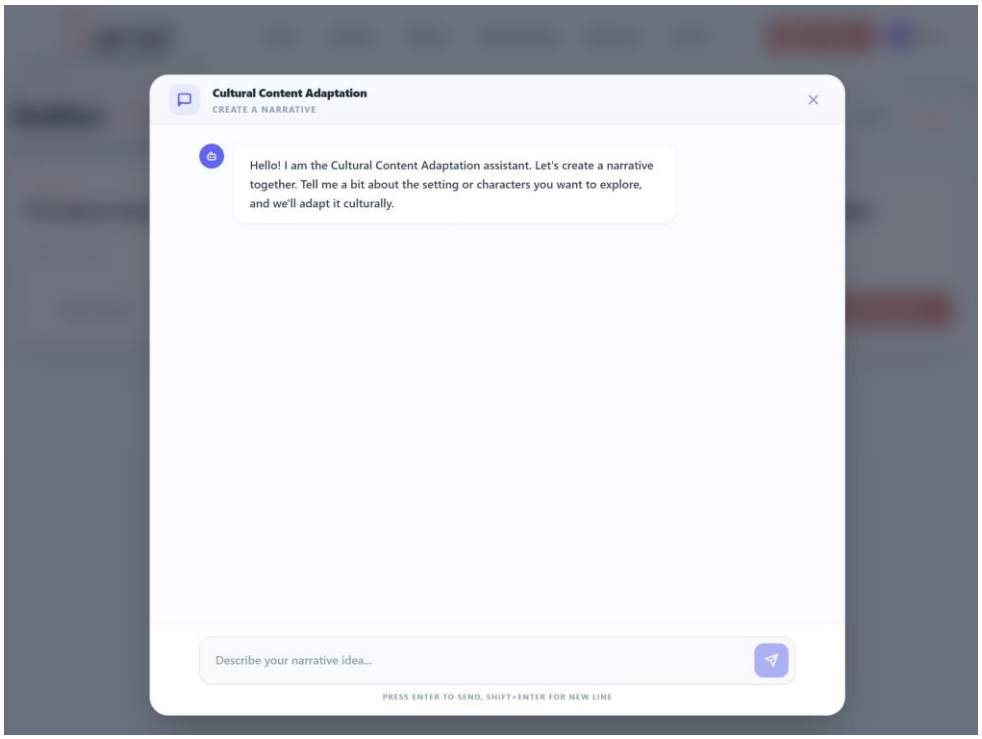

Figure 8 Cultural Content Adaptation Enabler page

The enabler is presented as a chat-based interactive interface, where the system acts as an intelligent assistant guiding the user through the narrative creation and adaptation process. Upon initialization, the assistant prompts the user to provide input regarding the desired setting, characters, or storyline. Based on this input, the system generates and adapts content while taking into account cultural context, semantics, and localisation aspects. The interface emphasizes natural language interaction, allowing users to iteratively refine their ideas through dialogue. The input field at the bottom enables users to describe their narrative concepts, while the system responds with structured suggestions or adapted outputs. This conversational paradigm simplifies complex AI operations, making them accessible to non-technical users and supporting creative exploration. The R1 interaction view supports the following functions:

- analyzes user-provided narrative elements,
- incorporates cultural and contextual knowledge,
- generates adapted narrative content aligned with specific domains or audiences, and
- supports iterative refinement through continuous user interaction.

3.3 Projects Page

Figure 9 presents the Project Workspace interface of the HAMLET CCH, which serves as the central R1 environment for organizing projects, managing project-related assets and supporting collaboration around creative workflows. This interface brings together project management, enabler access points and collaboration functionalities into a unified workspace tailored to specific application domains (e.g., Theatre, Dance, Games).

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

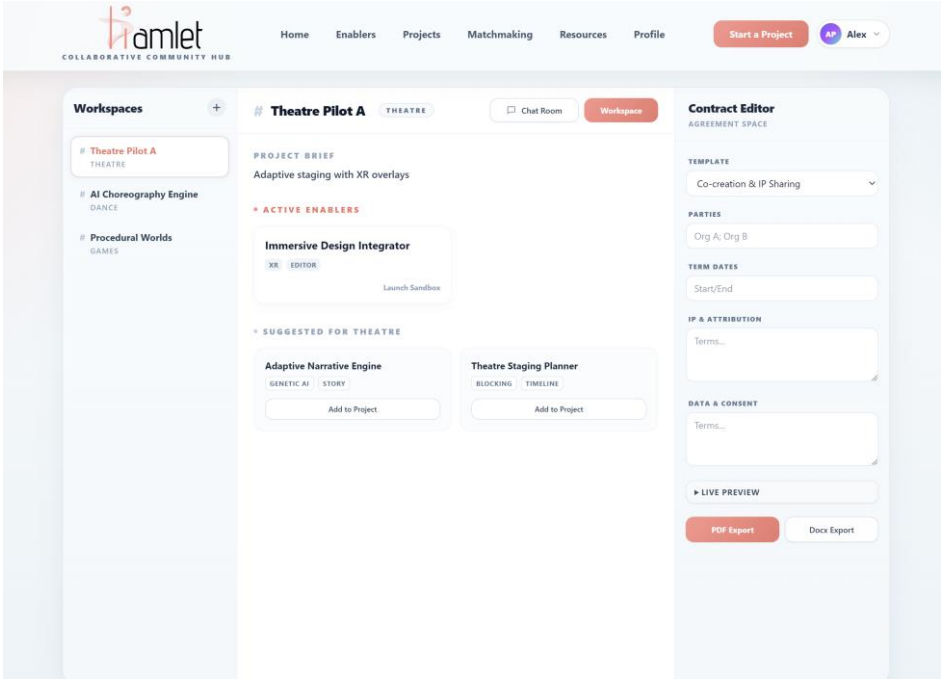

Figure 9 Projects Workspace Page

On the left-hand side, a workspace navigation panel provides access to multiple projects, allowing users to switch between different workspaces (e.g., Theatre Pilot A, AI Choreography Engine, Procedural Worlds). This supports the management of parallel creative workflows and facilitates domain-specific organization of activities. The central panel displays the selected project, in this case, “Theatre Pilot A”, including a project brief that outlines the creative objective (e.g., adaptive staging with XR overlays). Below this, the interface highlights the enablers associated with the project workspace, such as the Immersive Design Integrator, which can be accessed through the CCH interface according to the current R1 integration scope.

In addition, the system presents recommended enablers based on the project’s domain and context (e.g., Adaptive Narrative Engine, Theatre Staging Planner). These suggestions support the composition of project workspaces by allowing users to identify relevant AI tools for a given project context. The “Add to Project” functionality supports the association of enablers with the active project, reflecting the modular and composable nature of the HAMLET architecture.

At the top of the workspace, users can switch between collaborative modes, such as a chat-based interaction space (Chat Room) and the structured project workspace view. This dual-mode interaction supports both communication and workflow management within the same environment. On the right-hand side, the interface incorporates a Contract Editor, which introduces a governance and collaboration layer within the platform. In the R1 implementation, this component supports the definition of agreement-related information, including co-creation terms, intellectual property (IP) sharing, timelines, and data consent. Features such as template selection, party definition, and export options (e.g., PDF, DOCX) support the formalization of collaborative activities and improve transparency in multi-stakeholder projects.

Figure 10 presents the Collaborative Chatrooms interface of the HAMLET CCH, which supports real-time communication, coordination, and knowledge exchange among users participating in shared projects and workflows. This component plays a central role in enabling collaborative and co-creative interactions, complementing the platform’s AI-driven functionalities.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

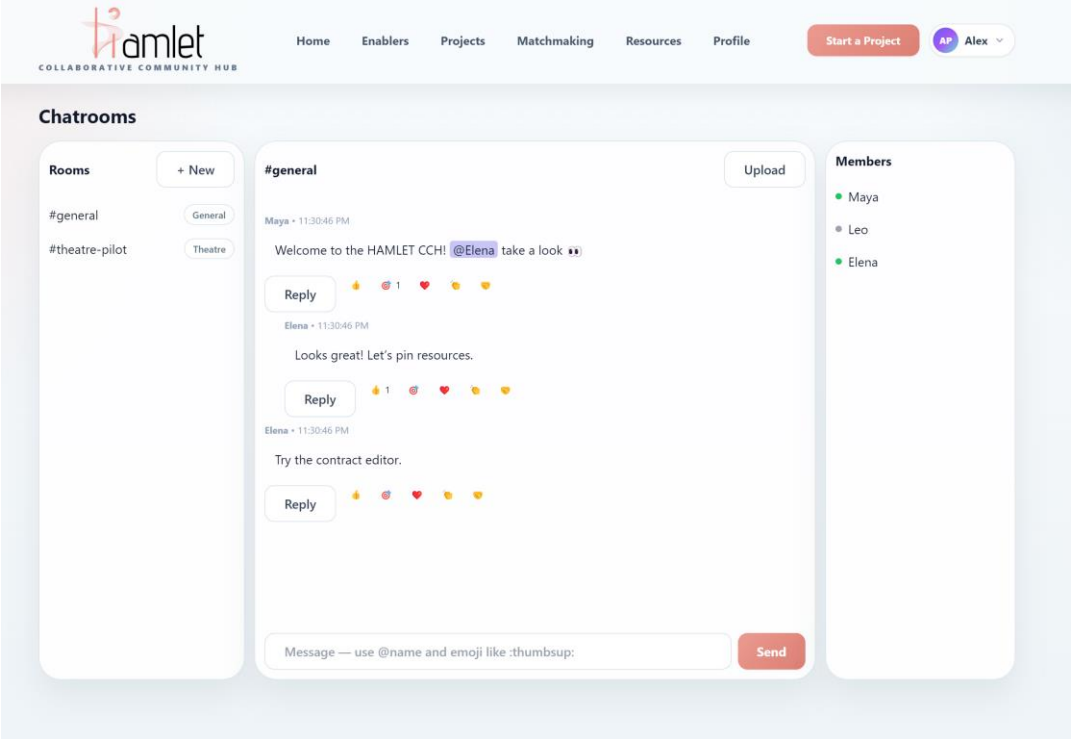

Figure 10 CCH Chatroom

The interface follows a three-panel layout, designed to facilitate efficient communication and contextual awareness:

- On the left-hand side, a Rooms panel lists available communication channels (e.g., #general, #theatre-pilot), allowing users to organize discussions based on projects, domains, or topics. The ability to create new rooms supports flexible structuring of collaborative spaces tailored to specific activities.
- The central panel displays the active conversation thread, where users can exchange messages in a structured, time-stamped format. The interface supports:
    - direct replies and threaded interactions,
    - user mentions enabling targeted communication, and
    - reaction-based feedback (e.g., emojis), enhancing engagement and informal interaction.

Additionally, functionalities such as file uploads and message input fields allow users to share resources, coordinate tasks, and discuss outputs generated by the platform's enablers. The chatroom functionality is also intended to support search and retrieval of previous messages and shared resources, helping users locate earlier discussions, uploaded material and relevant outputs as conversations grow over time. Additional refinements, such as media-focused views, pinned messages or other mechanisms for highlighting important exchanges, may be considered during R2 and pilot-facing refinement according to user needs.

- On the right-hand side, a **Members panel** provides visibility into active participants within the chatroom, supporting awareness of team composition and facilitating direct collaboration among users.

From a functional perspective, the Chatrooms interface enables:

- **real-time collaboration** across distributed users,
- **connection with project workspaces**, allowing users to discuss enabler outputs and creative processes, and

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

- **knowledge sharing and coordination**, supporting both technical and creative exchanges.

4 Backend Implementation

The HAMLET CCH backend provides the R1 implementation basis for the architecture described in Section 2 and exposes the functionalities surfaced by the UI in Section 3 through RESTful APIs, service-interaction logic, and a persistence layer built on relational, graph, and vector stores. The backend follows a microservices-oriented design, where domain concerns such as identity, projects, enablers, collaboration, and data are exposed through a unified API Gateway. Services share a common authentication context based on OAuth 2.0 / OpenID Connect tokens issued by the central Identity Provider (see Section 2.3), and rely on the common data substrate composed of the multimodal Data Lake, Knowledge Graph, and Vector Database (see Section 2.4).

The remainder of this section describes the principal API surfaces, the underlying data model, and the runtime logic that binds them together. The structures presented here reflect the R1 implementation status at the time of writing. They will be refined and extended in subsequent iterations as additional enablers and pilot requirements are integrated.

4.1 API Surfaces

The R1 backend exposes its functionality through a layered set of REST endpoints organized by domain, following established REST architectural principles [3]. All endpoints are versioned (e.g., /api/v1/...), return JSON payloads, and — with the exception of the public authentication endpoints — require a valid bearer token issued by the Identity Provider. The following tables summarise the principal endpoints grouped by functional area.

The endpoints in Table 4 define the access-control and user-lifecycle mechanisms of the HAMLET CCH. They provide the entry point through which users register, authenticate, maintain active sessions, recover access to their accounts, and manage their personal profile information. These APIs are directly linked to the login, registration, email verification, password reset, and user profile views of the frontend, while also providing the security context required by all other backend services. More specifically, it separates publicly accessible authentication operations from protected user-management operations. Public endpoints, such as /signup, /signin, /verify-email, and /password-reset, are available before a user is authenticated and support account creation and access recovery. In contrast, protected endpoints, such as /signout, /users/me, and /users/{id}, require a valid bearer token and are executed within the authenticated user context. This distinction ensures that only verified and authorised users can access personal information, update profile attributes, or retrieve user-related data. The sign-up endpoint creates a new user account by collecting basic registration information, such as email, password, display name, and the requested role. The account is not considered fully active until the email-verification flow is completed through a one-time verification token. This prevents unauthorised or invalid account creation and supports a more trustworthy user base within the CCH. The sign-in endpoint delegates credential validation to the central Identity Provider and returns access and refresh tokens, which are then used by the frontend to authenticate subsequent requests to the API Gateway. Session continuity is handled through the token-refresh and sign-out endpoints. The /auth/refresh endpoint allows the frontend to request a new short-lived access token using a valid refresh token, avoiding the need for repeated logins while preserving security. The /auth/sign-out endpoint invalidates the active session and revokes the refresh token, ensuring that access can be terminated explicitly when the user logs out or when a session must be closed for security reasons. The user-profile endpoints provide controlled access to user

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

information after authentication. The /users/me endpoint retrieves the profile, assigned roles, and user preferences of the currently authenticated user, enabling personalised views and role-aware UI behaviour. The corresponding PATCH /users/me endpoint allows users to update editable profile attributes, such as display name, avatar, organisation, and interface preferences. Finally, /users/{id} support the retrieval of public user profiles, subject to role-based access control, enabling collaboration features such as project membership, contributor visibility, and community interaction without exposing private information.

Table 4 Authentication and user management endpoints

Method	Endpoint	Description	Auth
POST	/api/v1/auth/sig-nup	Register a new user (email, password, display name, role request). Triggers email verification.	Public
POST	/api/v1/auth/sign in	Authenticate a user and issue access + refresh JWT to-kens.	Public
POST	/api/v1/auth/re-fresh	Exchange a valid refresh token for a new access token.	Public (refresh token)
POST	/api/v1/auth/si-gnout	Revoke the current session and invalidate the refresh token.	Bearer
POST	/api/v1/auth/ver-ify-email	Confirm an email address using a one-time verification token.	Public (token)
POST	/api/v1/auth/pas sword-reset	Initiate or complete a password-reset flow.	Public
GET	/api/v1/users/me	Retrieve the authenticated user's profile, roles, and preferences.	Bearer
PATCH	/api/v1/users/me	Update profile attributes (display name, avatar, organ-isation, preferences).	Bearer
GET	/api/v1/users/{id}	Retrieve a public profile for a given user (subject to RBAC).	Bearer

The endpoints in Table 5 define how projects are created, accessed, updated, populated with members, and connected with digital assets inside the HAMLET CCH. In the backend logic, a project acts as the main work-space where users collaborate around a specific creative, cultural, or pilot-oriented objective. Each project can group together its members, selected HAMLET enablers, uploaded or generated assets, related activity history, and the associated collaboration space. As a result, the project API provides the organisational layer that connects user identity, AI-enabled workflows, asset management, and community interaction. Moreo-ver, it distinguishes between general project-discovery operations and project-specific management opera-tions. The GET /api/v1/projects endpoint returns the projects visible to the authenticated user, supporting pagination and filtering by domain or pilot. This enables the frontend to populate dashboards, project lists,

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

and workspace overviews according to the user's permissions. The POST /api/v1/projects endpoint allows an authenticated user to create a new project by defining its title, brief, domain, visibility level, and initial members. This operation establishes the basic workspace structure and creates the initial project context used by the rest of the backend services. The project-detail endpoints provide access to the internal state of an existing workspace. The GET /api/v1/projects/{id} endpoint retrieves the project metadata together with information such as active enablers, members, recent activity, and potentially linked assets. Since this information is project-specific, access is restricted to authenticated users who are members of the project. The PATCH /api/v1/projects/{id} endpoint supports controlled updates to project metadata, status, or visibility, allowing project owners or editors to refine the workspace as the project evolves. This may include changing the project description, updating its operational status, or modifying whether the project is private, shared with selected users, or visible within a broader community context.

The DELETE /api/v1/projects/{id} endpoint implements a soft-delete mechanism rather than immediate permanent removal. This means that deleted projects are retained for auditability and can remain recoverable for a defined period, such as 30 days. This approach is important for collaborative environments, where projects may contain generated assets, user contributions, activity logs, and references to external repositories. Soft deletion therefore supports traceability, accidental-deletion recovery, and compliance with internal governance requirements. Membership management is handled through the project-member endpoints. The POST /api/v1/projects/{id}/members endpoint allows a project owner or editor to invite or add users to a project and assign them a specific role, such as owner, editor, contributor, or viewer. These roles determine what actions each member can perform within the workspace, including editing project information, launching enablers, uploading assets, or only viewing results. The DELETE /api/v1/projects/{id}/members/{userId} endpoint allows project owners to remove members when access is no longer required. Together, these endpoints provide the role-based collaboration model required for controlled multi-user workspaces. The asset-related endpoints connect each project with the HAMLET data-management layer. The GET /api/v1/projects/{id}/assets endpoint lists all assets associated with a project, including uploaded files, generated outputs, intermediate artefacts, or references stored in the Data Lake. The POST /api/v1/projects/{id}/assets endpoint enables project members to upload a new file or register an existing asset through a Data Lake URI. This design allows the backend to support both direct asset ingestion and references to already stored resources, while maintaining the project as the main point of access and organization.

Table 5 Project and workspace endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/projects	List projects visible to the authenticated user (paginated, filterable by domain/pilot).	Bearer
POST	/api/v1/projects	Create a new project (title, brief, domain, visibility, initial members).	Bearer
GET	/api/v1/projects/{id}	Retrieve project details including active enablers and recent activity.	Bearer + member
PATCH	/api/v1/projects/{id}	Update project metadata, status, or visibility.	Bearer + owner/editor

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

DELETE	/api/v1/projects/{id}	Soft-delete a project (retained for audit and recoverable for 30 days).	Bearer + owner
POST	/api/v1/projects/{id}/members	Invite or add a user to a project with a given role.	Bearer + owner/editor
DELETE	/api/v1/projects/{id}/members/{userId}	Remove a member from a project.	Bearer + owner
GET	/api/v1/projects/{id}/assets	List all assets produced or uploaded within the project.	Bearer + member
POST	/api/v1/projects/{id}/assets	Upload or register a new asset (multipart or Data Lake URI).	Bearer + member

In R1, retrieval of project material from the CCH is handled through the CCH API and database logic. Assets, generated outputs and related metadata are associated with the relevant project and user records, allowing authorised users to retrieve the material linked to a specific workspace through API requests based on the project ID and user ID. This provides the access route for obtaining project-level material from the Hub database where such retrieval is relevant to the specific enabler or pilot workflow. Since not all HAMLET enablers produce outputs intended for external production environments such as Unity, Unreal Engine or Blender, more dedicated export options will be addressed according to pilot needs during the R2 and pilot-facing integration work. These may include downloadable project packages, engine-oriented asset bundles, structured metadata exports or other workflow-specific formats where required.

The endpoints in Table 6 define the R1 API basis for discovering, inspecting, instantiating, attaching, executing, monitoring and terminating enabler instances. In the CCH backend, enablers represent the core technological capabilities to be exposed through the platform, including horizontal AI services, cultural-content generation tools, multimodal processing components, XR-related services, and domain-specific functionalities. The Enablers Catalogue therefore acts as a controlled registry of available tools, while the execution layer manages how these tools are launched and used either experimentally or within a concrete project workflow. It separates the enabler lifecycle into three main stages: discovery, deployment, and execution. The discovery endpoints allow users to browse and understand the available technological components before using them. The GET /api/v1/enablers endpoint returns the list of registered enablers and supports filtering by domain, tag, or capability. This enables the frontend catalogue view to present relevant tools depending on the user's needs, such as text-to-3D generation, cultural-content adaptation, avatar animation, visual analysis, or asset enhancement. The GET /api/v1/enablers/{id} endpoint retrieves the detailed descriptor of a selected enabler, including its expected inputs, outputs, configurable parameters, supported versions, execution requirements, and possible limitations. This descriptor is essential for both the UI and the backend orchestration layer, as it defines how the enabler can be invoked and integrated into workflows. The sandbox endpoint supports experimentation before full project integration. The POST /api/v1/enablers/{id}/sandbox endpoint launches an isolated instance of an enabler, allowing authenticated users to test its functionality without binding the results to a specific project. This is useful for exploratory use cases where users want to understand the behaviour of an enabler, test input formats, adjust parameters, or evaluate preliminary outputs.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Since sandbox instances are isolated, they reduce the risk of affecting shared project data or persistent workflows during experimentation. Project-bound enabler operations are handled through the /projects/{pid}/enablers/{id} endpoints. The POST /api/v1/projects/{pid}/enablers/{id} endpoint attaches a selected enabler to a project workflow, making it available within the project context and linking its execution to the project's members, assets, and activity history. This operation attaches an enabler from the catalogue to a workspace, preparing it for project-level use. In contrast, the DELETE /api/v1/projects/{pid}/enablers/{id} endpoint detaches the enabler from the project when it is no longer needed or when the workflow configuration changes. This distinction allows each project to maintain its own active set of tools according to its creative, technical, or pilot-specific requirements. The instance-execution endpoints manage the runtime behaviour of launched enablers. The POST /api/v1/instances/{instanceId}/run endpoint submits an execution request to a running enabler instance, together with the required input payload. Depending on the specific enabler, the input may include text prompts, images, 3D objects, metadata, configuration parameters, references to Data Lake assets, or outputs produced by previous enablers. This endpoint is therefore the main entry point for invoking HAMLET's AI and XR capabilities programmatically from the CCH.

Monitoring and lifecycle control are provided through the remaining instance endpoints. The GET /api/v1/instances/{instanceId} endpoint retrieves the current status of an enabler instance, including execution state, logs, progress information, errors, and resource usage. This information is important for long-running operations, such as 3D generation, video diffusion, asset reconstruction, or multimodal analysis, where users need feedback about progress and completion. The DELETE /api/v1/instances/{instanceId} endpoint terminates an active instance and releases the allocated computational resources. This prevents unused services from consuming infrastructure capacity and supports scalable resource management across multiple users and projects. Access control in Table 6 reflects the sensitivity of enabler execution. While catalogue browsing requires only bearer authentication, project-bound operations require project membership, and instance-level operations are restricted to the instance owner or authorised project users. This ensures that users can only execute, monitor, or terminate enabler instances for which they have appropriate permissions. It also prevents accidental interference with other users' workflows and supports traceability of all enabler executions.

Table 6 Enablers catalogue and execution endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/enablers	List available enablers, filterable by domain, tag, or capability.	Bearer
GET	/api/v1/enablers/{id}	Retrieve the descriptor of a given enabler (inputs, outputs, parameters, version).	Bearer
POST	/api/v1/enablers/{id}/sandbox	Launch an isolated sandbox instance of the enabler for experimentation.	Bearer
POST	/api/v1/projects/{pid}/enablers/{id}	Attach an enabler instance to a project workflow.	Bearer + project member

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

DELETE	/api/v1/projects/{pid}/enablers/{id}	Detach an enabler instance from a project.	Bearer + owner/editor
POST	/api/v1/instances/{instanceId}/run	Submit an execution request to a running enabler instance with input payload.	Bearer + instance owner
GET	/api/v1/instances/{instanceId}	Retrieve the status, logs, and resource usage of an instance.	Bearer + instance owner
DELETE	/api/v1/instances/{instanceId}	Terminate an enabler instance and release allocated resources.	Bearer + instance owner

Table 7 defines the collaboration layer of the CCH, supporting project communication, message exchange, room membership, and lightweight interaction feature such as reactions and mentions. This functionality is provided through a real-time messaging service that combines REST APIs for room and history management with a WebSocket channel for live communication. The REST endpoints manage the persistent aspects of collaboration. The GET /api/v1/rooms endpoint retrieves the chat rooms available to the authenticated user, allowing the frontend to display personal, project-related, or team-based communication spaces. The POST /api/v1/rooms endpoint creates a new room, which may either be a general collaboration space or be explicitly linked to a project. In the latter case, the room becomes part of the project workspace, enabling members to discuss tasks, enabler outputs, generated assets, and workflow decisions in a shared context. Message history and posting are handled through the room-message endpoints. The GET /api/v1/rooms/{id}/messages endpoint retrieves previous messages with pagination and time-range filtering, ensuring that long conversations can be loaded efficiently without overloading the client. The POST /api/v1/rooms/{id}/messages endpoint allows room members to send new messages containing text, mentions, and attachments. Attachments may include references to project assets, generated outputs, or supporting files, thereby connecting communication with the wider HAMLET data and project-management layers.

Table 7 Collaboration and messaging endpoints

Method	Endpoint	Description	Auth
GET	/api/v1/rooms	List chat rooms the user belongs to.	Bearer
POST	/api/v1/rooms	Create a new chat room (optionally linked to a project).	Bearer
GET	/api/v1/rooms/{id}/messages	Retrieve message history with pagination and time-range filters.	Bearer + member
POST	/api/v1/rooms/{id}/messages	Post a new message (text, mentions, attachments).	Bearer + member

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

POST	/api/v1/rooms/{id}/members	Add a member to a chat room.	Bearer + admin
POST	/api/v1/messages/{id}/reactions	Add or remove an emoji reaction to a message.	Bearer + member

4.2 Data Model

The HAMLET CCH backend uses three complementary persistence mechanisms to store and retrieve platform information according to its intended use. The relational database, implemented in PostgreSQL [8], stores the operational state of the platform, including users, projects, memberships, enabler descriptors, runtime instances, chat messages, and asset records. This layer is responsible for transactional consistency, access-control enforcement, and efficient querying of structured platform data. In parallel, semantic relationships between platform entities are projected into the Knowledge Graph. This allows the system to represent higher-level links, such as which user created a project, which enabler produced a specific asset, which assets are derived from previous inputs, or which workflows are associated with a particular cultural-content objective. These semantic links support provenance tracking, cross-modal discovery, and reasoning across users, projects, tools, and generated outputs. The Vector Database complements these two stores by managing high-dimensional embeddings extracted from textual, visual, audio, and multimodal assets. These embeddings enable similarity search, content recommendation, asset retrieval, and matchmaking between user needs and available enablers or resources. For example, a text prompt, image, or cultural asset can be embedded and compared with existing project materials or enabler capabilities to support more intelligent search and reuse. The following tables summarize the main relational entities used by the backend. They focus on the core attributes of each entity, while implementation-specific elements such as foreign keys, indexes, audit logs, soft-delete flags, and timestamp-management details are omitted for readability.

In R1, the CCH stores project information, membership information, enabler-instance records and asset records through the backend data model. A project acts as the main workspace reference: when a project is created, the backend creates the relevant project record, associated collaboration space and membership records, and when the project is opened the service retrieves the project details, members, associated enabler instances and recent assets. Data produced or used by enablers is linked to this project structure through enabler-instance and asset records. When an enabler produces an output, the output is stored in the multimodal Data Lake and the corresponding asset record keeps the provenance links to the producing enabler instance, project and user. In this way, the CCH connects project-level information, enabler executions and generated or uploaded resources through traceable backend records.

Table 8 describes the users entity, which stores the registered accounts of the CCH. Each record represents a platform user and provides the identity information required for authentication, authorisation, personalisation, and collaboration. The id field is a UUID primary key used internally to reference the user across projects, memberships, assets, chat rooms, and enabler executions. The email field stores the user's primary email address and is defined as unique, ensuring that each account can be reliably identified during sign-in and notification workflows. The password hash field stores the securely hashed password using Argon2. When a user authenticates through a federated identity provider, this field may remain null, since credential verification is delegated externally through the Identity Provider. The display name and organisation fields

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

support the user-facing profile information shown across the CCH. The display name is used in project membership lists, chat rooms, activity feeds, and asset ownership views, while the organisation field can indicate the user's affiliated institution, company, pilot partner, or creative organization. The roles field stores the user's role-based access-control assignments, such as admin, creator, developer, or user. These roles determine the level of access a user has across the platform, including whether they can manage enablers, create projects, administer users, or participate as a standard collaborator. The status field records the lifecycle state of the account, distinguishing between active users, accounts pending email verification, and suspended accounts. Finally, the "created at" and "updated at" timestamps provide basic lifecycle tracking for each account. They support auditability, account-management workflows, and operational monitoring by recording when a user account was created and when its information was last modified.

Table 8 Users - Registered platform accounts

Column	Type	Description
id	UUID (PK)	Internal user identifier.
email	VARCHAR (unique)	Primary email address; used for sign-in and notifications.
password_hash	VARCHAR	Argon2 hash; null when the user authenticates via a federated identity provider.
display_name	VARCHAR	Public name shown across the Hub.
organisation	VARCHAR	Affiliated institution or company (optional).
roles	TEXT[]	Assigned RBAC roles (e.g., admin, creator, developer, user).
status	ENUM	active, pending_verification, suspended.
created_at/updated_at	TIMESTAMP	Lifecycle timestamps.

Table 9 describes the projects entity, which represents the main workspace structure of the CCH. Each project acts as an organisational container for users, enablers, assets, chat activity, and creative workflows. The id field is the internal UUID primary key used to reference the project across memberships, asset records, enabler instances, activity logs, and collaboration services. The title field stores the project name displayed in the workspace, dashboards, project lists, and navigation views. The brief field provides a free-text description of the project's creative or technical objective. This description helps users understand the purpose of the workspace and can also support search, recommendation, or future matching with relevant HAMLET enablers. The domain field classifies the project according to the pilot or application area, such as theatre, dance, games, or other. This enables filtering, reporting, and domain-specific organisation of projects. It also allows the backend to associate projects with relevant enablers, templates, workflows, or pilot requirements. The visibility field controls how broadly the project can be accessed or discovered. A private project is restricted to its owner or explicitly authorised users, "members_only" projects are accessible to assigned members, while public projects may be visible more broadly within the CCH, depending on the platform's access-control

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

rules. Ownership and collaboration links are captured through the “owner_id” and “chatroom_id” fields. The “owner_id” references the user who created the project and acts as the primary owner for administrative operations, such as updating metadata, managing visibility, or deleting the project. The “chatroom_id” links the project to its associated collaboration room, allowing project discussions, messages, and shared activity to remain connected to the workspace. The status field records the lifecycle state of the project. Active projects are currently available for use, archived projects are retained for reference but no longer actively edited, and deleted projects are soft-deleted according to the backend retention policy. Finally, the “created_at” and “updated_at” timestamps provide lifecycle tracking, supporting auditability, project monitoring, and chronological sorting in the user interface.

Table 9 Projects - Collaborative workspaces

Column	Type	Description
id	UUID (PK)	Internal project identifier.
title	VARCHAR	Project title displayed in the workspace.
brief	TEXT	Free-text description of the creative objective.
domain	ENUM	Pilot domain: theatre, dance, games, other.
visibility	ENUM	private, members_only, public.
owner_id	UUID (FK → users.id)	Project creator and primary owner.
chatroom_id	UUID (FK → rooms.id)	Associated chat room.
status	ENUM	active, archived, deleted.
created_at / updated_at	TIMESTAMP	Lifecycle timestamps.

Table 10 describes the “project_memberships” entity, which manages the relationship between users and projects. While the projects table defines the workspace itself, the membership table specifies which users belong to each project and what permissions they have inside that workspace. It therefore provides the basis for project-level access control, collaboration, and role-based operations. The “project_id” field references the corresponding project in the projects table, while the “user_id” field references the participating user in the user’s table. Together, these two fields form a composite primary key, ensuring that each user can only have one membership record per project. This prevents duplicated memberships and provides a clear mapping between users and their assigned project roles. The role field defines the user’s permissions within the project. An owner typically has full administrative control, including managing members, updating project metadata, changing visibility, and deleting or archiving the project. An editor can contribute to the workspace by modifying project content, uploading assets, launching or configuring enablers, and participating in workflow activities. A viewer has read-only access, allowing the user to inspect project information, assets, and outputs without modifying the workspace. The “joined_at” field records when the user became part of the project. This timestamp supports auditability, project history, and membership tracking, while also enabling the frontend to display team composition and collaboration timelines.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Table 10 "Project_members" - Project membership and roles

Column	Type	Description
project_id	UUID (FK, PK)	Reference to projects.id.
user_id	UUID (FK, PK)	Reference to users.id.
role	ENUM	Project-level role: owner, editor, viewer.
joined_at	TIMESTAMP	Timestamp of joining the project.

Table 11 describes the enablers entity, which stores the catalogue-level information for each technological component available through the CCH. Each record represents one registered enabler, including its identity, functional classification, execution metadata, version, and operational status. This table therefore acts as the backend registry that allows enablers to be discovered, described, selected, and launched within project workflows. The id field is the internal UUID primary key used to uniquely identify each enabler across the platform. It is referenced when users browse the catalogue, attach an enabler to a project, or launch an executable instance. The name field provides the human-readable display name shown in the frontend, such as Adaptive Narrative Enabler, and helps users understand the purpose of the tool. The category field distinguishes between horizontal and industry-specific enablers. Horizontal enablers provide generic capabilities that can be reused across several creative and cultural domains, while industry-specific enablers target more specialised requirements linked to areas such as theatre, dance, games, or other pilot contexts. This classification supports filtering, catalogue organisation, and future reporting on the coverage of HAMLET technologies. The "domain_tags" and "capability_tags" fields provide a more flexible description of each enabler. Domain tags indicate the application areas where the enabler is relevant, such as theatre, dance, games, or analytics. Capability tags describe the technical function of the enabler, such as XR, motion analysis, diffusion-based generation, semantic enrichment, multimodal processing, or content adaptation. These tags enable search, filtering, recommendation, and matchmaking between project needs and available tools.

The descriptor field is stored as JSONB and contains the structured technical specification of the enabler. This may include the expected input and output schemas, configurable parameters, supported file types, execution constraints, resource requirements, and integration details. Using JSONB allows each enabler to expose a flexible descriptor while still keeping the information queryable by the backend. This is important because different enablers may require different types of inputs, such as text prompts, images, 3D models, motion signals, metadata, or references to Data Lake assets. The "image_ref" field stores the container image reference used to launch executable instances of the enabler. This connects the catalogue entry with the deployment layer and allows the execution service to instantiate the correct software component in a sandbox or project-bound environment. The version field records the semantic version of the enabler, supporting reproducibility, compatibility checks, and controlled updates over time. Finally, the status field indicates whether the enabler is currently available, deprecated, or experimental. Available enablers can be used in active workflows, deprecated enablers remain visible for compatibility or historical reasons but may be replaced in future versions, and experimental enablers are under evaluation or limited testing.

Table 11 Enablers - Registry of available AI tools

Column	Type	Description

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

id	UUID (PK)	Internal enabler identifier.
name	VARCHAR	Display name (e.g., Adaptive Narrative Enabler).
category	ENUM	horizontal or industry_specific.
domain_tags	TEXT[]	Associated domains (theatre, dance, games, analytics, …).
capability_tags	TEXT[]	Functional tags (XR, motion, diffusion, semantics, …).
descriptor	JSONB	Input/output schema, parameter spec, resource profile.
image_ref	VARCHAR	Container image reference used to launch instances.
version	VARCHAR	Semantic version of the enabler.
status	ENUM	available, deprecated, experimental.

Table 12 includes records for both active and historical executions of HAMLET enablers. While the enablers table describes the available tools at catalogue level, this table captures each concrete runtime instance launched by a user, either for sandbox experimentation or as part of a project workflow. The id field uniquely identifies each execution instance, while "enabler_id" links the instance to the corresponding registered enabler. The "project_id" field connects the instance to a specific project when the enabler is executed within a workspace; it remains null when the instance is launched in sandbox mode. This distinction allows the backend to separate exploratory executions from project-bound workflows. The "owner_id" field identifies the user who launched the instance and is used for access control, monitoring, and accountability. The mode field indicates whether the execution belongs to a sandbox or project context, while the status field tracks the current lifecycle state of the instance, such as pending, running, completed, failed, or terminated. The config field stores the resolved execution configuration in JSONB format, including selected parameters, input references, model options, runtime settings, and other enabler-specific options. This supports reproducibility, debugging, and later inspection of how a given output was generated. The "resource_handle" field links the logical instance record to the underlying compute resource, such as a Kubernetes pod [10], job identifier, or external execution service reference. Finally, the "started_at" and "ended_at" timestamps capture the execution timeline, enabling progress tracking, runtime analysis, auditability, and resource-usage monitoring.

Table 12 "Enabler_instances" - Running and historical executions

Column	Type	Description
Id	UUID (PK)	Internal instance identifier.
enabler_id	UUID (FK → enablers.id)	Enabler being executed.
project_id	UUID (FK → projects.id, null)	Owning project; null for sandbox instances.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

owner_id	UUID (FK → users.id)	User who launched the instance.
mode	ENUM	sandbox or project.
status	ENUM	pending, running, completed, failed, terminated.
config	JSONB	Resolved configuration and parameters.
resource_handle	VARCHAR	Reference to the underlying compute resource (e.g., Kubernetes pod, job ID).
started_at / ended_at	TIMESTAMP	Execution lifecycle timestamps.

In addition to the core entities described above, the R1 backend uses auxiliary structures for assets, collaboration spaces and chat messages. These structures are referenced in the API surfaces and backend service-interaction flows described in Sections 4.1 and 4.3. They support the management of uploaded and generated resources, the operation of project-based communication spaces, the persistence of chat history, and the recording of accountability-relevant backend actions.

Table 13: Assets – Uploaded and generated resources

Column	Type	Description
id	UUID (PK)	Internal asset identifier.
project_id	UUID (FK → projects.id)	Project associated with the asset.
owner_id	UUID (FK → users.id)	User who uploaded or generated the asset.
producing_instance_id	UUID (FK → enabler_instances.id, null)	Enabler instance that produced the asset, where applicable.
modality	ENUM	Asset type, e.g., text, image, audio, video, motion, 3D, or metadata.
data_lake_uri	VARCHAR	Location of the underlying object in the multimodal Data Lake.
metadata	JSONB	Descriptive metadata, provenance information, usage context, and asset-specific attributes.
created_at / updated_at	TIMESTAMP	Lifecycle timestamps.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

The assets entity links uploaded or generated resources to the relevant project, user, producing enabler instance, and Data Lake location. This supports traceability of creative outputs and enables generated assets to be reused, referenced, or further processed within project workflows.

Table 14: Rooms – Collaboration spaces

Column	Type	Description
id	UUID (PK)	Internal room identifier.
project_id	UUID (FK → projects.id, null)	Linked project, where the room is project-specific.
name	VARCHAR	Room name displayed in the chat interface.
type	ENUM	Room type, e.g., general, project, or private.
created_by	UUID (FK → users.id)	User who created the room.
created_at/updated_at	TIMESTAMP	Lifecycle timestamps.

The rooms entity supports the collaboration layer of the CCH by defining the communication spaces available to users. Rooms may be general or linked to a specific project, allowing discussions to remain connected to the relevant workspace and project context.

Table 15: Messages – Chatroom messages

Column	Type	Description
id	UUID (PK)	Internal message identifier.
room_id	UUID (FK → rooms.id)	Room where the message was posted.
sender_id	UUID (FK → users.id)	User who posted the message.
parent_message_id	UUID (FK → messages.id, null)	Parent message for threaded replies, where applicable.
content	TEXT	Message text.
attachments	JSONB	Linked files, project assets, generated outputs, or supporting resources.
mentions	UUID[]	Referenced users.
created_at/updated_at	TIMESTAMP	Lifecycle timestamps.

The messages entity persists communication history within each room. It supports threaded replies, user mentions, and attachments, allowing users to discuss project activities, enabler outputs, generated assets, and workflow decisions in a shared context.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

4.3 Backend Logic and Service Interactions

The backend services interact through synchronous REST calls for request-response operations and through an internal event bus for asynchronous notifications such as instance state changes, asset ingestion completion, and chat events. The API Gateway enforces authentication, rate limiting, and request routing; downstream services trust the identity claims carried in the validated JWT and apply RBAC policies locally based on the user's global roles and project-level membership. The following paragraphs describe the principal R1 backend flows, with cross-references to the data model presented above.

Sign-up and sign-in: When a user submits the registration form, the Auth service validates the payload, ensures email uniqueness, hashes the password with Argon2 (in line with current password-storage recommendations [6], [7]), inserts a new row in users with status "pending_verification", and dispatches a verification email containing a signed, single-use token. Confirmation of that token transitions the account to active and provisions the default RBAC role. On sign-in, the service verifies credentials (or delegates to a federated identity provider in the SSO scenario described in Section 2.3), issues a short-lived access JWT and a longer-lived refresh token, and records the session in a refresh-token table that supports server-side revocation. Subsequent calls present the access token in the Authorization header; the API Gateway validates its signature and expiry, extracts identity and role claims, and forwards them to downstream services in a trusted internal header.

Project creation and membership: Creating a "project" result in three coordinated operations carried out within a single transaction in the Project service: insertion of a new row in projects with the requesting user as owner, creation of an associated chat room in rooms, and insertion of the owner row in "project_members" with role owner. Subsequent invitations add rows in "project_members" with the appropriate role; the Auth context ensures that only owners and editors may modify membership. When a project is opened in the workspace UI, the service retrieves the project details, members, associated enabler instances, and recent assets needed to display the Projects Page (Section 3.3).

Enabler discovery and execution: The Enablers Catalogue is populated from the enablers registry; filtering and search rely on the "domain_tags" and "capability_tags" columns, combined with semantic-similarity queries against the Vector Database when the user issues a natural-language query through the Hub Assistant. In the R1 backend design, launching a sandbox or project-bound execution follows four main steps:

- the Enabler service validates the request against the enabler descriptor and the user's permissions;
- it creates a new row in "enabler_instances" with status pending and a resolved configuration;
- it submits a job to the orchestrator (Kubernetes-based), which schedules a container from "image_ref" and reports back through the event bus;
- as the job progresses, the service transitions the instance through running and finally completed or failed, persisting any produced assets through the Data Layer described below.

Asset ingestion and propagation: When an enabler instance produces an output, the binary payload is uploaded to the multimodal Data Lake and a corresponding row is inserted in the assets table with provenance pointers to the producing instance, project, and user. An ingestion worker computes modality-appropriate embeddings (text, image, audio, motion) and writes them to the Vector Database keyed by the asset id. In parallel, the Knowledge Graph is updated with new nodes and edges reflecting the provenance chain. This propagation makes newly produced assets available for similarity search, recommendation functions, and provenance-aware queries across the platform.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

Collaboration: Chat messages are persisted in messages and broadcast to room members through the WebSocket channel; mentions trigger notification events on the internal bus, which the Notification service consumes to update unread counters and, optionally, dispatch email digests. Reactions and threaded replies are stored as auxiliary rows referencing the parent message, supporting the threaded interaction patterns described in Section 3.3.

Cross-cutting concerns: Every state-changing call writes an entry to the audit log, capturing the actor, action, target entity, and a hash of the request payload, in support of accountability and reproducibility requirements. Rate limiting is applied at the API Gateway on a per-user and per-IP basis; circuit breakers protect downstream services from cascading failures when an enabler or external system becomes unresponsive. Observability is provided through structured logs, distributed traces propagated via the standard W3C trace context headers [9], and metrics exported to a central monitoring stack, ensuring that the platform's operational behaviour can be analysed and refined throughout the pilots.

The auxiliary entities described in Section 4.2 complement these service flows by linking generated or uploaded assets to projects, users, enabler instances, and Data Lake locations, while also supporting project-based rooms and message histories. Together with refresh-token and audit-log structures, these records provide the backend basis for traceability, collaboration, session management, and accountability across the CCH.

5 Conclusion

5.1 Integration Activities

This deliverable has presented the first release of the HAMLET CCH, covering the overall architecture (Section 2), the frontend implementation of the Home Dashboard, Enablers Catalogue and Projects pages (Section 3), and the backend implementation that exposes the platform's APIs, persists its state and supports interactions between users, enablers and the HAMLET Data Lakes (Section 4). Together, these components form an R1 implementation baseline that supports user onboarding, project collaboration, enabler discovery, selected enabler interaction flows, and provenance-aware asset management. The Hub is designed as the central access point through which researchers, creators and developers will progressively engage with the HAMLET ecosystem throughout the remainder of the project. The implementation choices documented above have been guided by the requirements gathered in WP1 and by the integration needs of the technical work packages.

Integration activities from the perspective of the CCH focus on establishing the Hub as the connective tissue between the platform's users and the HAMLET technical components delivered by the rest of the consortium. Key coordination efforts are organized along the following pillars:

Orchestrated Enabler Execution via the Enabler Integration Agent: The Hub backend does not invoke enablers directly. Launch requests issued through the Enablers Catalogue or the Projects page are routed to the Enabler Integration Agent (Task 2.4), which acts as the Tool Orchestrator and is responsible for parameter resolution, scheduling and lifecycle management. The Hub persists the resulting "enabler_instances" records and surfaces their state in the UI, ensuring a clean separation between user-facing workflows and the underlying execution machinery. Identity and Authorisation Alignment: the Hub provides the authoritative user, role and project-membership records consumed by the rest of the ecosystem. Coordination activities ensure that the JWT-based identity claims issued by the Auth service (Section 4.1) and the RBAC model documented

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

in Section 2.3 are honoured uniformly by enablers and by the agentic framework, so that every action performed on a user's behalf can be authenticated and authorised against a single source of truth.

Bidirectional Data Lake Interaction: the Hub is both a producer and a consumer of content held in the HAMLET Data Lakes. Coordination activities enable it to: Ingest assets generated by enablers and register them in the assets table together with provenance pointers to the producing instance, project and user; Query the multimodal and vector stores to drive semantic search, recommendation and matchmaking features across the Home Dashboard, Enablers Catalogue and Projects pages.

5.2 Next Steps

As an immediate next step after R1, controlled consortium access will be prepared for selected internal users, including relevant technical partners and pilot-facing consortium representatives. This first access route will focus on the R1 platform components described in this deliverable: the Home Dashboard, Enablers Catalogue, the enabler interfaces currently included in R1, Projects page, chatroom functionality, authentication/API structures, backend data model and asset-management logic. The purpose of this access phase is to allow partners to check navigation, account access, enabler discovery, project workspace logic, collaboration flows and the currently available Hub/enabler interactions before broader pilot-facing use.

The first controlled-access version will be used as a structured handover from R1 implementation to R2 integration. As such, it will not expose all final HAMLET workflows at once. Further enabler connections, multi-enabler orchestration, complete end-to-end workflows, pilot-specific adaptations, export/download refinements, responsible-use guidance and broader user validation will be consolidated during the R2 transition and subsequent WP5 activities. Local or offline execution of specific enablers, such as the Adaptive Narrative Enabler, is not part of the general R1 CCH access model and will be assessed per enabler during R2 and pilot-facing integration, taking into account technical packaging, dependencies, licensing, compute requirements, data access and pilot needs. Feedback gathered through controlled access will be used to prioritize the next integration steps and to prepare the CCH for more mature pilot-facing validation.

Therefore, the next phase of work on the CCH is organized around five activities that will take the platform from its current R1 implementation baseline towards a more mature, validated and pilot-ready service.

Controlled deployment and access preparation: Building on the first controlled-access route described above, the CCH will be further stabilised for consortium and pilot-facing use. This step includes stabilizing the deployment environment, hardening the API Gateway and authentication services, reviewing access-control, legal and accessibility requirements, and preparing onboarding documentation for the consortium and pilot partners. The public or wider access route will be defined once the required stability, security, legal and accessibility checks have been completed.

User testing and interface refinement: Once the platform is available to selected internal and pilot-facing users, key UI and backend behaviours will be exercised through controlled testing activities. Candidate areas include the clarity of the Home Dashboard, navigation through the Enablers Catalogue, the usability of the Projects page, the wording and pacing of the sign-up and onboarding flows, and the effectiveness of search and recommendation functions. Where appropriate, alternative interface options may be compared with users to support iterative design decisions. Testing will remain proportionate to the maturity of the R1 implementation and will focus on improving usability, clarity and task completion.

Refinements during the first pilot round: The first pilot round will be the principal source of feedback for refining the CCH. Quantitative signals from platform use and monitoring activities will be combined with

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

qualitative feedback gathered through ad-hoc discussions and usability sessions with pilots, feedback forms, and TAA-supported audience/user feedback activities where relevant. The resulting findings will drive targeted refinements to the API surfaces described in Section 4.1, the data model described in Section 4.2, the backend service flows described in Section 4.3, and the UI patterns documented in Section 3.

Enhancements: In parallel with pilot-driven refinements, the CCH will be extended with additional capabilities that were not fully covered in the R1 release. Planned enhancements include richer collaboration features in the Projects page, such as threaded discussions, notifications and activity streams; deeper integration with the agentic framework; more expressive search and recommendation experiences powered by the Vector Database and the Knowledge Graph; and additional administrative tooling for project and enabler governance.

Bug fixing and stabilisation: A continuous bug-fixing and stabilisation track will run alongside the activities above. Issues reported by pilot users, consortium partners and monitoring activities will be triaged, prioritised and addressed through the project's development workflow. Where relevant, regression checks will be added for confirmed defects so that the overall quality, reliability and integration maturity of the platform increases steadily over the course of the pilots.

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362

References

[1] HAMLET Consortium, Deliverable D1.3 - Architecture, Technical Specs & MockUps.R1, 2025.

[2] HAMLET Consortium, Deliverable D1.2 - Application Scenarios and Use Cases Definitions, 2025.

[3] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD dissertation, University of California, Irvine, 2000.

[4] D. Hardt (Ed.), The OAuth 2.0 Authorization Framework, IETF RFC 6749, October 2012. https://www.rfc-editor.org/rfc/rfc6749

[5] M. Jones, J. Bradley and N. Sakimura, JSON Web Token (JWT), IETF RFC 7519, May 2015. https://www.rfc-editor.org/rfc/rfc7519

[6] A. Biryukov, D. Dinu, D. Khovratovich and S. Josefsson, Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications, IETF RFC 9106, September 2021. https://www.rfc-editor.org/rfc/rfc9106

[7] OWASP Foundation, Password Storage Cheat Sheet, 2024. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[8] The PostgreSQL Global Development Group, PostgreSQL Documentation, 2025. https://www.postgresql.org/docs/

[9] W3C, Trace Context, W3C Recommendation, 23 November 2021. https://www.w3.org/TR/trace-context/

[10] The Kubernetes Authors, Kubernetes Documentation, 2025. https://kubernetes.io/docs/

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101178362