Electronic, didactic and innovative platform for learning based on multimedia assets

e-DIPLOMA

Funded by the European Union

D.4.2 Platform software

Version No. V1.3

28 February 2024

HISTORY OF CHANGES			
Version*	**Publication date**	**Beneficiaries**	**Changes**
V1.0	24.01.2024	BME	▪ Initial version of Deliverable Owner
v1.1	14.02.2024	BME, TUD, BRA	▪ First draft version
v1.2	20.02.2024	BME, TUD, BRA	▪ Pre-final version for internal review
v1.3	28.02.2034	BME, TUD, BRA	▪ Final version for submission

(*) According to the section "Review and Submission of Deliverables" of the Project Handbook

1 Technical References

Project Number	101061424
Project Acronym	e-DIPLOMA
Project Title	Electronic, Didactic and Innovative Platform for Learning based On Multimedia Assets
Granting Authority	European Research Executive Agency (REA)
Call	HORIZON-CL2-2021-TRANSFORMATIONS-01
Topic	HORIZON-CL2-2021-TRANSFORMATIONS-01-05
Type of the Action	HORIZON Research and Innovation Actions
Duration	1 September 2022 – 31 August 2025 (36 months)
Entry into force of the Grant	1 September 2022
Project Coordinator	Inmaculada Remolar Quintana

Deliverable No.	D4.1: Platform technology specification
Work Package	WP4: Development of training modules and e-Platform Deployment
Task	T4.1: Analysis of remote e-learning in teacher training.
Dissemination level*	PU- Public
Type of license:	CC-BY
Lead beneficiary	Budapesti Műszaki és Gazdaságtudományi Egyetem (BME)
PIC of the Lead beneficiary	999904228
Contributing beneficiary/ies	Technische Universiteit Delft (TU Delft) Brainstorm Multimedia SL (BRAINSTORM)
PIC of the Contributing beneficiary/ies	TU Delft: 999977366 BRAINSTORM: 999441732
Due date of deliverable	29. 02. 2024
Actual submission date	29. 02. 2024

2 Table of Contents

2.1 List of Figures

2.2 List of tables

3 Introduction

3.1 Executive Summary

This document is the documentation component of deliverable D4.2 of e-DIPLOMA project, which accompanies the software constituting the essence of this deliverable. The software is a technical solution for a cloud-based service, integrated with a learning management system, that allows for the delivery and execution of applications implementing web-based, virtual reality, augmented reality, web-conferencing based, and other multi-user, interactive and experiential learning applications.

The document summarises results of T4.2: *Creation of the cloud resource management, network synchronisation, and avatar-based interaction for virtual rooms* and T4.3: *Creation of the extended reality tools, network streaming, audio-video interaction framework for the e-platform, and a Video Conferencing - Virtual Reality Learning System.*

3.2 Relation to Other Project Documents

This document contains the API specification for the platform, following the requirements identified in

D4.1 E-platform specification.

The ability to integrate applications capable of network synchronisation, avatar-based interaction for virtual rooms (T4.2), as well as extended reality tools, network streaming, audio-video interaction framework for the e-platform, and video conferencing (T4.3) are demonstrated in this deliverable with stub applications. The development of actual game applications is **not** part of this deliverable, as they will be part of T4.4, T4.5, T4.6, and T4.7, and thus will be presented in *D4.3 Prototypes*.

3.3 Abbreviation List

Among the acronyms used most often in the present document are, in alphabetical order, the following:

AI: Artificial Intelligence

API: Application Programming Interface

AR: Augmented Reality

AWS: Amazon Web Services

EC2: (Amazon) Elastic Compute Cloud

eLTI: Enable Learning Tools Integration

GDPR: General Data Protection Regulation

GUI: Graphical user interface

ID: Identity

LDS: Learning Design System

LMS: Learning Management System

LO:	Learning Object
LTI:	Learning Tools Integration
NLP:	Natural Language Processing
OAuth:	Open Authorization
OEM:	Original Equipment Manufacturer
OIDC:	OpenID Connect
REST:	REpresentational State Transfer
S3:	(Amazon) Simple Storage Service
SAML:	Security Assertion Markup Language
SNS:	(Amazon) Simple Notification Service
SQL:	Standard Query Language
UE:	Unreal Engine
VM:	Virtual Machine
VR:	Virtual Reality

3.4 Reference Documents

See the References Section included in this document.

4 Functionality

4.1 Summary of Requirements based on the e-Platform specification

The e-DIPLOMA Platform is a framework for cloud-based management of interactive educational computer applications, related assets and education materials, as well as a means for distributing and executing those applications on user and cloud devices with lobby-based matchmaking for group activities, and monitoring activities for research and profiling purposes.

The e-DIPLOMA platform must be **compatible with existing systems** used in education to the highest degree possible, but also bring high-end **gaming, virtual and augmented reality technologies** into the framework.

The platform must be **extensible**: it should be possible to add new assets and applications indefinitely. It should also be **modular**: constituting components should communicate through well-defined and, where applicable, standard interfaces. The platform should be **documented** on every level ranging from development to user functionality, with **examples** provided to demonstrate how to integrate new pieces of software or infrastructure. While the platform should be cloud-based, and thus relying on a cloud service to operate, it should be **portable** in the sense that existing hardware resources can also be employed.

The platform must support the **evaluation of activities** performed within it. Therefore, it must provide a mechanism for the **storage and management of acquired data**, including server activity logs and client-side measurements. It has to manage the granting and revoking of **consent**, and must **store data** in a safe and secure way in accordance with the consent given. Users need to have well-defined **roles**, with permission to perform certain actions tied to those roles.

The platform must provide a **repository for assets**, including stock assets for learning object authoring, user assets for **co-creation**, and assignment submission.

The cloud infrastructure should be **re-deployable** to other cloud service accounts, facilitating later use by e-DIPLOMA project partners of adopting institutes. This has to be achieved using **infrastructure-as-code** facilities.

The **cost** of cloud services should be controllable by the instructors and managers operating the system. **Installation** on user devices should be simple, and **updates** should be automatic.

The platform should enable the use of game engines, integrating applications that allow for:

- Real-time graphics.
- Real-time physical simulation.
- VR visualisation techniques.
- VR navigation techniques.
- AR solutions.
- Real-time societal and civil infrastructure simulation.
- Procedural content generation.
- Content generation by AI.
- Behavioural AI.

Note that these are not implemented as part of the platform service, but by later applications developed for the platform, targeting specific prototypes. Much of the above capabilities are based on the game engines chosen as the basis of those applications. However, the platform is shipped with technology demonstrator applications, exemplifying the integration of game engines into the platform as well as the use of the above technologies.

4.2 Underlying technologies and services

The platform uses the **Moodle** learning management system. The e-DIPLOMA project has a Moodle server installed at the premises of **BME**. It is available at http://cg.iit.bme.hu:3004. Every institute adopting the platform can use their own Moodle system, managing their own students and teachers according to their standards. Any authentication method supported by Google, preferred by or previously implemented by the institute, can be used. This includes **OIDC** and **SAML** federated authentication.

The e-DIPLOMA service is implemented in **AWS**, hosted as a set of web pages on the cloud, communicating with other AWS services for authentication (**Cognito**), data storage (**S3**), serverless computations (**API Gateway**, **Lambda**), game server operation (**GameLift**) and Elastic Computing System (EC2).

Moodle is a certified LTI 1.3 platform. The e-DIPLOMA service integrates with Moodle using **LTI**, where the service takes the role of an LTI tool. Any Moodle installation can be connected to e-DIPLOMA by configuring the LTI connection on both ends.

LTI is based on **OIDC** authentication. **AWS Cognito** is used as the OIDC identity provider towards the e-DIPLOMA service. Cognito is configured to incorporate, as federated identity providers, Moodle tool registrations. This is done through the **eLTI** library. Therefore, e-DIPLOMA users keep their identities and roles defined by their institutional Moodle accounts, and need only to log in to Moodle once.

The e-DIPLOMA Frontend, i.e. the web pages providing the user interface for e-DIPLOMA users, is a set of static **HTML5** web pages developed in the **Angular** framework in **Typescript**. They are served to users from publicly readable **AWS S3** buckets.

The e-DIPLOMA Backend is a set of serverless operations (**AWS Lambda**) accessible through websocket and http endpoints (**AWS API Gateway**). They store data in the **AWS DynamoDB** database, and rely on **S3** buckets for file storage.

The e-DIPLOMA Launcher is a **.NET 6.0** desktop application written in C#. It makes use of both **Windows Forms** (for single instance functionality) and **WPF** for the user interface. It can be installed on a Windows computer using an **MSI** installer.

AWS EC2 is a service that allows the use of computer instances in the cloud. Applications, and Brainstorm's Edison in particular, can be deployed to **EC2** instances directly.

AWS GameLift is a managed way to run game servers on EC2 instances. e-DIPLOMA applications based on a client-server game engine model run their game servers on GameLift.

Unity and **Unreal Engine** are game engines used to implement some of the e-DIPLOMA applications.

NICE DCV is a remote desktop solution that allows accessing EC2 instances through a graphical desktop interface. It allows multiple users to join the same computer, and for a user to share their web camera stream.

Microsoft Teams is a video conferencing solution integrated with Brainstorm's Edison.

4.3 Definitions of terms

activity

An item visible on a Moodle course page, added by a Teacher (acting in the role of e-DIPLOMA Course Administrator). e-DIPLOMA activities are a specific type of activity. Clicking such an activity opens an e-DIPLOMA Frontend webpage. e-DIPLOMA activities execute some *learning objects*.

application

A software product available through the e-DIPLOMA platform, using which educational materials and activities can be realised. There can be multiple *versions* of an application.

asset

A collection of files managed by the e-DIPLOMA platform. Assets can be part of *learning objects*, but they can also be private files of users, the result of work or co-creation performed by users, activity logs or measurement data.

client

In a multiuser application, the client is the program that is executed on the user's device, responsible for display and interaction. Clients connect to a *server*.

cloud

Computer resources available through the network, allocated and paid for according to customer needs.

consent form

A collection of permissions to list, view, execute, contribute to, edit, or delete *resources,* including user data, measurement, activity logs, *learning objects*, *applications*, and *media assets*. A consent form lists the person(s) responsible for the appropriate handling of the data with their contacts, purposes and risk associated with the data use, and the users receiving the authorization through the consent form. Data owners can grant their consent to the terms of the consent form pertaining to any of their data *resources* individually, or revoke that consent.

course

A collection of *learning objects* designed to be studied in succession, teaching some subject.

external tool

A Moodle term for applications that are integrated into Moodle using LTI. e-DIPLOMA is an external tool for Moodle.

learning object

A unit of learning material, based on a *version* of an *application*, accompanied with associated *assets*. An activity always concerns a *learning object*.

LMS

Learning Management System, i.e. Moodle.

LTI Platform

The *LMS* in the LTI model, i.e. Moodle.

LTI Tool

The service that integrates into the *LMS* using LTI, i.e. the e-DIPLOMA platform.

module

One lesson's worth of learning material in a *course*. Consists of one of more *learning objects*.

OIDC identity provider

A server that provides authentication facilities via standard endpoints, e.g. providing digitally signed tokens that can be used to establish the provenance of messages sent between computers on the network. An identity provider can rely on another identity provider. Moodle works as an identity provider through LTI. Backed by LTI, Amazon Cognito is an identity provider for the platform front and backends, with user ID sources from Moodle accounts.

pilot

A sequence of classes teaching the learning material of one of the e-DIPLOMA *prototypes*. Pilots measure and document student and teacher activity using disruptive technologies in education, in order to arrive at conclusions about their use.

prototype

A *course* designed for e-DIPLOMA *pilots*, instrumented to measure and record user characteristics and actions for analysis and research.

resource

Any object persistently stored in the e-DIPLOMA system, including *courses*, *learning objects*, *applications*, *builds*, *media assets*, *asset groups*, and *consent forms*.

server

In a multiuser application, the server is the program that is executed on a dedicated computer, often in the *cloud*. The server is responsible for coherent simulation of virtual reality. *Clients* connect to a server.

service

An application providing functionality over the network.

version

A specific build of an *application*. *Learning objects* are authored on the basis of a specific application version.

4.4 Roles

A ***role*** is a set of capabilities or privileges assigned to a user. A user may be assigned multiple roles at the same time. e-DIPLOMA uses Moodle for managing user roles. The additional responsibilities related to the platform, namely those of the Application Developer, Game Engine Integrator, and Cloud Administrator, are also discussed as roles in this document, but they do not require access to Moodle or the e-DIPLOMA frontend. Instead they would work separately on their software, and interface with AWS using the AWS Console or the CLI, authenticating themselves with their AWS credentials.

Table 1: Roles relationships in e-DIPLOMA platform implementation.

e-DIPLOMA role	e-DIPLOMA Lobby symbol	Moodle role	LTI role	LTI claim
Student	♙	Student	Learner	http://purl.imsglobal.org/vocab/lis/v2/membership/Learner
Supervisor	♘	Non-editing teacher	TeachingAssistant	http://purl.imsglobal.org/vocab/lis/v2/membership/Instructor#TeachingAssistant
Moodle Administrator	♔	Site Administrator	Administrator	http://purl.imsglobal.org/vocab/lis/v2/system/person#Administrator
Course manager	♗	Teacher	Instructor	http://purl.imsglobal.org/vocab/lis/v2/membership#Instructor
Learning Object Author		e-DIPLOMA Learning Object Author	ContentDeveloper	http://purl.imsglobal.org/vocab/lis/v2/membership#ContentDeveloper
Application Deployer		e-DIPLOMA Application Deployer	ExternalContentExpert	ttp://purl.imsglobal.org/vocab/lis/v2/membership/ContentDeveloper#ExternalContentExpert
Application Developer		N/A	N/A	N/A
Game Engine Integrator		N/A	N/A	N/A
Cloud Administrator		N/A	N/A	N/A

A ***student*** is a user enrolled in an LMS course as a student. Students participate in e-DIPLOMA activities. Most often, students are the primary users of e-DIPLOMA Apps.

A ***supervisor*** is an educator who oversees e-DIPLOMA activities. Supervisors may join students in virtual environments, monitor student activity, and they have control over the use of cloud services.

The ***Moodle administrator*** is the person responsible for setting up and configuring Moodle at an institute. They are required to configure the e-DIPLOMA service as an LTI Tool.

A ***course manager*** is an educator responsible for populating a Moodle course with content, including e-DIPLOMA activities.

A ***learning object author*** is a developer of educational content, who pairs an e-DIPLOMA application with media assets and scripts to produce a piece of unique educational material, which can be delivered to students as an activity.

An ***application deployer*** is a person who uploads an application build to the e-DIPLOMA platform.

An ***application developer*** is a programmer who writes programs that use game engines, making use of a code stub that provides the necessary communication with e-DIPLOMA platform.

A ***game engine integrator*** creates a code stub for a new game engine or similar application environment, to ensure integration with the e-DIPLOMA platform.

A ***cloud administrator*** deploys e-DIPLOMA as a cloud service, and is able to monitor and control costs.

4.5 Platform overview

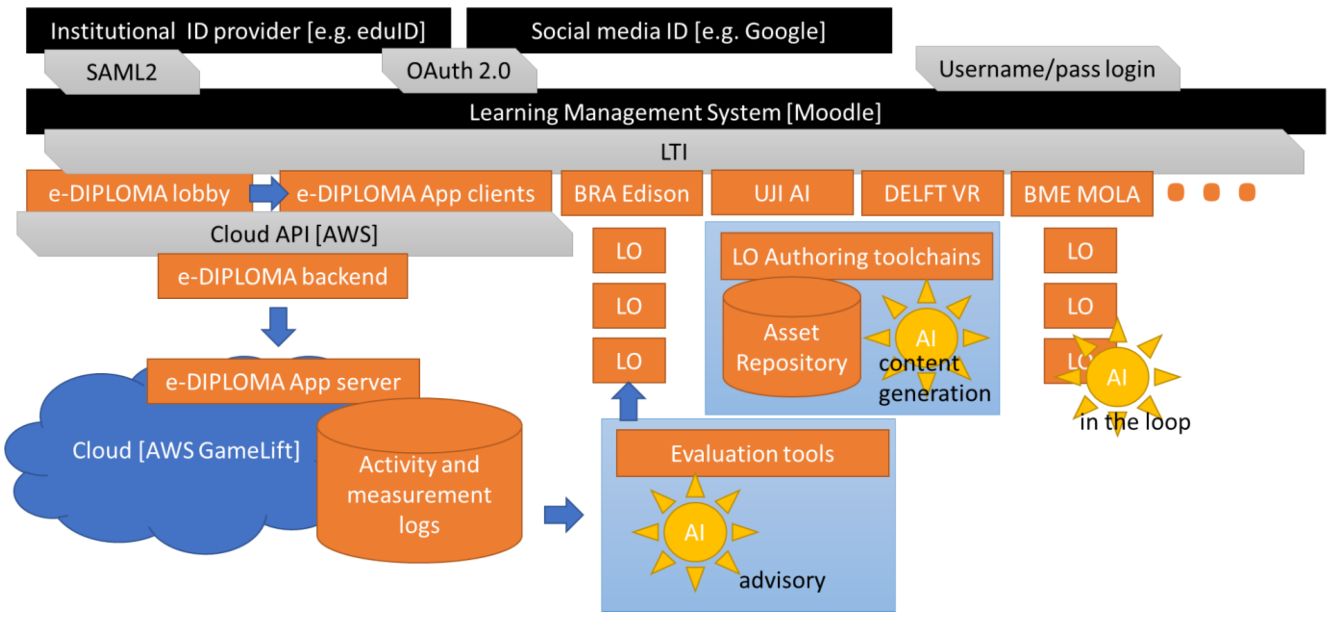


Figure 1: The e-DIPLOMA Platform with its environment and application features from D4.1

Figure 1 shows the overview of the e-DIPLOMA platform as drafted in the specification in deliverable D4.1. Black boxes symbolize pre-existing infrastructure, key interfacing standards are in grey, and orange boxes are high-level components of the platform. Users log in to *Moodle* using the method preferred in their institutes. Often, this means using a global service (e.g., log in with Google, or login with Facebook), or an institute-wide solution (e.g., EduID [22]). These services are based on the OAuth 2.0 or SAML 2.0 protocols. Alternatively, Moodle may be configured to allow other ways of logging in, like registering usernames and passwords only for Moodle. Once users are logged in to Moodle, they can access courses according to their roles. External Tool activities can be added to course pages by content editors or teachers and launched by students. The *e-DIPLOMA Lobby* is such a Tool. Moodle and the e-DIPLOMA Lobby are interfacing using the LTI protocol. The e-DIPLOMA Lobby is a web service for launching *Learning Objects* (LOs), including organizing events or sessions where multiple users take part. While some LOs may be implemented as web services themselves, the need for using high-end technologies like AR, VR, and gamification would typically require specialized desktop applications built on extensive middleware, and game engines in particular. Practically, a single application should be used to implement many LOs. Typical game AI like pathfinding may be accompanied by disruptive AI like chatbots to provide a responsive and immersive experience. It is foreseen that e-DIPLOMA partners contributing to WP4 each implement one or a few *applications*, and several LOs within each application, as part of the prototype course development process. In order for the development process to be replicable by adopters in the exploitation phase of the project, creation methodologies and toolchains must be documented and provided along with the applications. Procedural or AI-assisted content generation may be employed. For multiplayer scenarios (which includes not only group work but also real-time teacher-student interaction), application instances running on user devices must be connected to create a synchronized game world. This is provided by the *application server* component. Depending on the type of the LO, multiple simultaneous instances of the game world need to be simulated, accommodating interacting groups of students and teachers. These instances may be run on on-premise servers, but typically they need to be hosted in the AWS cloud. Launching these dedicated server computers in the cloud, managing the authentication of users joining the sessions, linking client applications to the server application, and monitoring resource usage is the task of the e-DIPLOMA Platform Backend, which is an AWS service. One could consider the e-DIPLOMA Lobby and the e-DIPLOMA Platform Backend to be a single, AWS-based service, with the lobby serving as the matchmaking interface for the backend. Finally, the server applications may record user activity relevant to academic or scientific evaluation. They can store this data in the cloud through the backend.

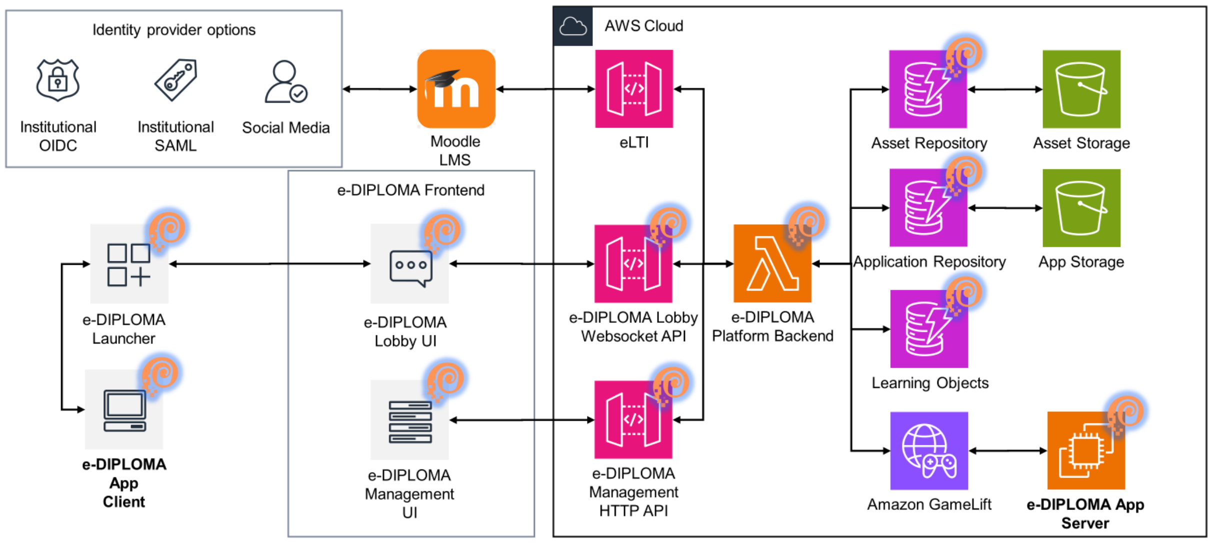


Figure 2: The e-DIPLOMA Platform core as implemented

Figure 4.2 gives an overview of the e-DIPLOMA platform components as implemented in the actual software. There are no conceptual differences, but the figure details strict platform functionality as opposed to application functionality. The Management UI and API are new elements, which are web pages with underlying infrastructure to create and manage e-DIPLOMA resources, including *applications*, *courses*, *learning objects*, *consent forms*, and *assets*. The e-DIPLOMA Launcher is an essential element that is installed on the client computers, managing the download and setup of application clients to these devices, and also launching and connecting application client instances to the servers in the cloud. In Figure 2, the entire chain facilitating the connections between the app clients and app servers can be followed.

4.5.1 Moodle

A Moodle server is maintained by BME for the purposes of the project, including administering the pilots. Educational institutes that adopt e-DIPLOMA can use their own LMS to manage their users and perform conventional course management activities. The e-DIPLOMA Platform is integrated into Moodle as an LTI-compliant *tool*. It is registered as a site-wide tool [10], making arduous manual configuration unnecessary when inserting e-DIPLOMA activities into courses. The course prototypes of e-DIPLOMA will be created as Moodle courses, the pilots will manage student evaluation through Moodle.

4.5.2 Lobby

The Lobby is the e-DIPLOMA platform interface that most users, and students in particular, are going to encounter most often. Activities on the Moodle course page launch this interactive web page, where students and teachers can see users logged into e-DIPLOMA through clicking the same activity in the LMS. They can communicate via chat, organize groups if necessary, and launch an application presenting the Learning Object associated with the activity. The service performs matchmaking, gathering the users willing to join the same interactive virtual environment.

The e-DIPLOMA Lobby is implemented with a web frontend and an AWS service backend, incorporating the eLTI (Enable LTI) [5] service for LTI compliance. Lobby usage and functionality is explained in detail in section 5.4.

4.5.3 Launcher

The Lobby must start desktop applications on client computers. This is only possible for applications installed and registered in the operating system. Therefore, the Lobby is accompanied by a Launcher component, similar to the launchers employed by Steam, GOG Galaxy, Epic Games, EA, Ubisoft, or Battle.net. The Launcher is responsible for keeping the installed e-DIPLOMA applications up-to-date and starting them when required by the lobby. Only the Launcher has to be registered with the operating system for browser access. Interaction with the Launcher is described as part of the Lobby experience in section 5.4, while technical details can be found in section 8.6.

4.5.4 Lobby API and lobby backend

A part of the backend is responsible for managing the lobby connections of several students and teachers, facilitating communication between them, handling their actions like joining or leaving virtual rooms, and, most importantly, to launch game servers in the cloud via GameLift. User clients can connect to these servers. This functionality is accessed from the browser running the Lobby page through a Websocket API. Websockets provide a two-way channel, meaning that the backend can send notifications to Lobby pages any time. In particular, actions by other students or teachers can be shown immediately on other user's computers.

4.5.5 **Authentication**

Every message sent from the browser to the backend has to be accompanied by proof of identity of the caller. Otherwise, anyone could claim access to user resources, posing a severe security risk. In the OICD protocol, such a proof is called an *access token*. The access token can be obtained from an OICD *identity provider*, and the backend can verify against the public cryptographic keys of the provider if they were indeed sent by the authorized user. These identity providers can rely on other providers, possibly establishing a long chain to an ultimate trusted entity that has reliably identified the user. In case of the e-DIPLOMA, the immediate identity provider for the Lobby is an AWS Cognito User Pool. The Lobby obtains the access token from Cognito via https at the endpoint URL that belongs to the pool. Cognito only provides the access token if authentication identifying the user has been performed beforehand. This happens when an e-DIPLOMA activity is started in Moodle. This is the role of eLTI (Enable-LTI, where LTI stands for the Learning Tools Interoperability standard). When a lobby activity is started, eLTI authenticates against Moodle (acting as the LTI Platform here). The Lobby web page is invoked with a code that can be traded for the access token with Cognito. For this to work, eLTI must be configured to work as an identity provider for the Cognito user pool (see section 6.4). Authenticating with Moodle can be based on user/password identification (as is the case with the e-DIPLOMA Moodle server), but most educational entities have an institution-wide single-sign-on solution based on either SAML or OICD. OICD identification is also supported by various social media platforms (log in with Google, log in with Facebook options). Moodle supports these options. Thus, a potential authorization chain could look like:

- eDiploma-pool in Cognito
- eLTI
- Moodle
- SAML single-sign-on solution, e.g. eduID at BME

4.5.6 **Applications**

Applications are developed by e-DIPLOMA partners as part of WP4, while creating the prototype courses. Later adopters may also develop applications, although that is not expected from most users of e-DIPLOMA.

Some applications may be implemented as web services. Others, being desktop applications, have to be installable and updatable through the e-DIPLOMA Launcher. They must work in close concert with the Web Lobby component and the backend in general for session management, authentication, and grading. They can access eLTI functionality through the backend.

Most applications are expected to work in a client-server model. The server is made ready to be launched by the backend on dedicated server instances in the cloud.

It is the responsibility of the server to record activity data and store it in the repository through the backend. Clients can gather measurement data about the user. It is the responsibility of the client to store this data into the repository through the backend.

Applications are deployed to the e-DIPLOMA platform through the Application Management Page (see section 5.9), which communicates with the backend through HTTP endpoints.

4.5.7 Asset repository

The asset repository is a multi-purpose data storage facility. Besides storing and organising media assets useful for the creation of interactive educational material, it also serves as personal data storage space for students and educators. Measurement and activity log data required for analysis are also stored as assets. Access to assets is governed by consent of their owners. All assets are accessible from the Asset Management Page (see section 5.3), which is part of the frontend. It communicates with the backend through HTTP endpoints.

4.5.8 Learning objects

A learning object combines an application version from the Application repository with assets from the Asset repository. What assets can be added and how they are interpreted depends on the application, but one of the assets is designated as the *root asset*, which tells the application what other assets to use. Its format may depend on the application, with possibilities including static JSON files or executable scripts in Lua or Python. Learning objects can be composed on the Learning Object Management Page (see 5.8), which communicates with the backend through HTTP endpoints.

4.5.9 Shared Immersive Assets Repository

In order to promote the employment of virtual production tools to generate immersive educational, an innovative case of asset repository will be provided in e-Diploma platform. The object is a general repository containing immersive content, such as 3D models and derived content like virtual environments, AR, XR assets to facilitate the creation of relatable educational content. To enable this possibility a set of learning objects components will be provided to the platform. The virtual reality experiences employ 3D models that can be reused to create educational content during the course, such exams, materials or presentations using the virtual production tools available in the platform.

5 User interface

5.1 System requirements

The client computer should run the Windows 10 (or later) operating system with a web browser. Chrome version 121.0.6167.185 has been tested most extensively. The user must have administrator privileges while installing the e-DIPLOMA Launcher, which can happen the first time a user enters the Lobby screen of the computer, or by redistributing the MSI installer directly. After that, if the launcher has been installed to a path accessible to the user, no such privileges are required, as application clients can be set up in standalone mode. System requirements for running the individual application may vary, but both Unity and Unreal engine based applications require a GPU-equipped desktop or laptop computer. VR applications can be extremely computationally intensive and may require a strong computer. Applications using thin clients like Brainstorm's Edison require the resources to run those thin clients, e.g. Nice DCV [24], which may be significantly less demanding.

5.2 Frontend pages

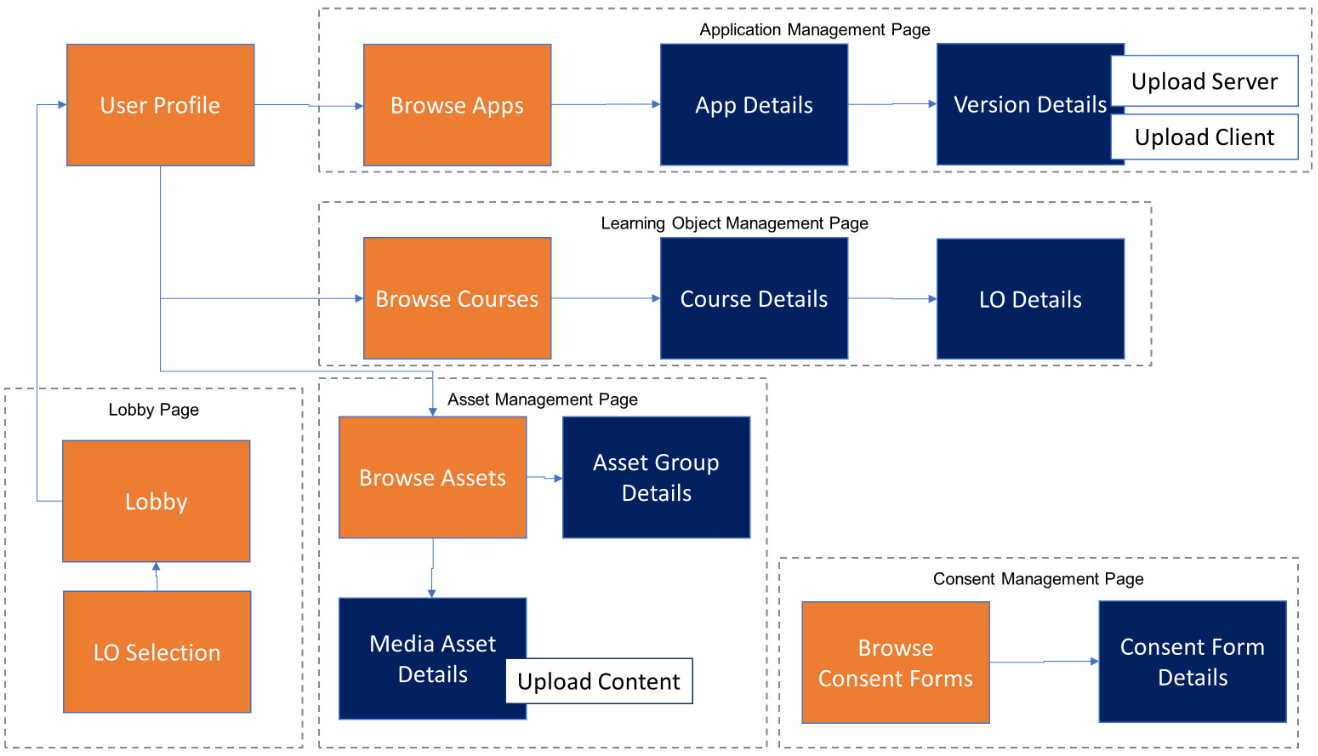


Figure 3: Conceptual overview of User Interface pages

Figure 3 provides a functional overview of the User Interface. The UI consists of web pages, all communicating with the backend through the AWS Gateway. The management pages are dedicated to the creation and editing of assets and asset groups, courses and learning objects, applications and versions. All of them can have consent forms attached, detailing what privileges other users are granted over the resources. Consent forms themselves can be edited in the Consent Forms Management Page. The Lobby page provides the interactive interface to distribute applications, organize group activities and launch educational sessions.

5.3 Asset Management Page

Every user of the platform can act as an *asset owner*. They can upload files (3D models, textures, images, videos, presentations, etc.) to the platform, typically accompanied by a description and a thumbnail image – creating a *media asset*. Recordings made about the actions performed, i.e. activity logs, are also assets. Similarly, measurements made are stored as assets. Every user is the *owner* of the assets they created. In case of multi-participant activity logs, the person who initiated the session (typically a supervising teacher) is designated as the asset owner.

Users can also create *asset groups*, and add *media assets* or other *asset groups* to them – producing arbitrary groupings of assets. The same asset can be in multiple groups, and groups can include multiple groups containing the same asset at some level of depth. It is even legal, if ill-advised, to create circular groupings, with e.g. group A containing group B and group B containing group A.

All *media assets* and *asset groups* are initially *private*, meaning that they only appear in listings for the user who created them, i.e. the *owner* of the asset. Owners of assets can enrol their assets under consent forms, thereby granting their permission to the kind of use specified by the consent form, to the users specified in the consent form, under the responsibility of the people specified in the consent form. Through these granted permissions, assets other than the user's own assets can show up in listings, and can be used in appropriate ways. Consent can be granted for entirely unlimited public use, publishing the assets to the community.

Private assets are intended for either

- development use (e.g. while creating and testing a new learning object, its assets may be private, and made accessible when the learning object is considered ready for teaching), or
- as a personal storage space for users, especially instructors and students. Learning objects may allow the use of these resources, selected and accessed by the participants when the educational activity is performed, not when the learning object is authored. A presentation delivered in a virtual environment or a 3D model created by a student can be such an asset.

Asset management can be accessed through the Learning Management System. The user should log in to the LMS, and then click on an e-DIPLOMA activity in any course. This activity may be one specifically posted for the purposes of accessing asset management, or another activity for any purpose. In the latter case, the user interface includes an Asset Management button that redirects the user to the Asset Management Page.

The Asset Management Page consists of the following elements:

- on the left side,
    - a filter field where you can enter text to display only those assets and asset groups that match the entered text,
    - a dropdown to select the minimum privilege level of the assets listed (list, view, execute, contribute, edit, delete, and owned),
    - a listing of the assets (both media and group types) matching current the filter and privilege level — these allow the selection and deletion of the assets,
    - buttons to create a new asset or asset group.

- On the right side, details of the currently selected asset or asset group are displayed. Selection is possible by clicking the asset or asset group on the left side. If a new asset or asset group is to be created, its properties can also be given here. For media assets, controls include:
    - text field for the title,
    - text field for the description,
    - a file upload area that accepts files or folders. These are automatically packed in a zip file for upload,
    - a list of Consent forms under the permissions of which the asset is accessible to non-owners,
    - Save and Cancel buttons to finalise or discard edits. Editing is only possible for users with at least *edit* privileges on the asset.
      For asset groups, the controls are similar, but instead of the upload area, there is
    - a list of included assets and asset groups. Here, assets can be added by dragging them from the left side listing and dropping them here. Conversely, dragging items out removes them from the group. These are only possible if the user has at least *contributes* access privilege to the asset group.
- Editing an asset group means that its associated assets will be listed, allowing deletion/editing. When adding a new asset, it will be linked to the edited group by default.
- Creating/editing a media asset means that the file can be uploaded, the thumbnail can be uploaded, and the description can be entered.

Learning object authors should create an asset group for every learning object they create. Supervisors and students should create, for every educational activity, an asset group for assets they want to access from within e-DIPLOMA Apps during that activity.

5.3.1 Consent management

Not only assets, but also applications, courses, learning objects need a control mechanism on access rights. The system that the e-DIPLOMA platform utilises is based on the concept that data owners need to give their informed consent on the use of their data. This is accomplished through consent forms. A consent form lists the person(s) responsible for the appropriate handling of the data with their contacts, purposes and risk associated with the data use, and the users receiving the authorization through the consent form. A user can be granted the following levels of access, each including the privileges of the preceding levels:

- list: the name and owner of the resource can appear in listings shown to the user
- view: the detailed properties of the resource can be shown to the user
- execute: the resource can be downloaded and uploaded and executed (if executable) or processed by an application
- contribute: if the resource is a collection (as a course is a collection of learning objects, and application is a collection of builds, and an asset group is a collection of assets), the user can add new elements to it
- edit: the user can change any properties
- delete: the user can remove the resource from the repository

Data owners can grant their consent to the terms of the consent form pertaining to any of their data resources individually, or revoke that consent (below).

5.4 Lobby Page (student role)

A student is a user enrolled in an LMS course as a student. The corresponding role in Moodle is *Student*. They are able to access the course page in the LMS. They can access the e-DIPLOMA platform if there has been an e-DIPLOMA activity added to the course page by a course supervisor.

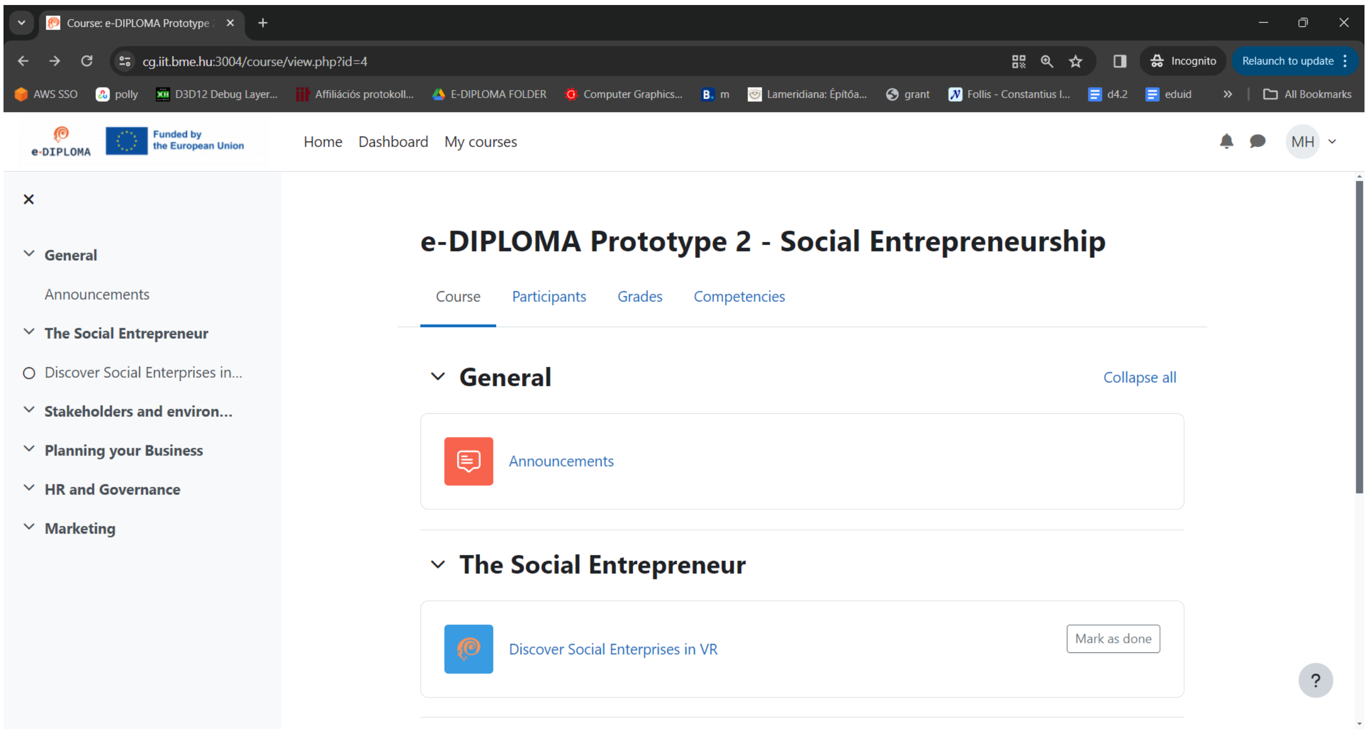
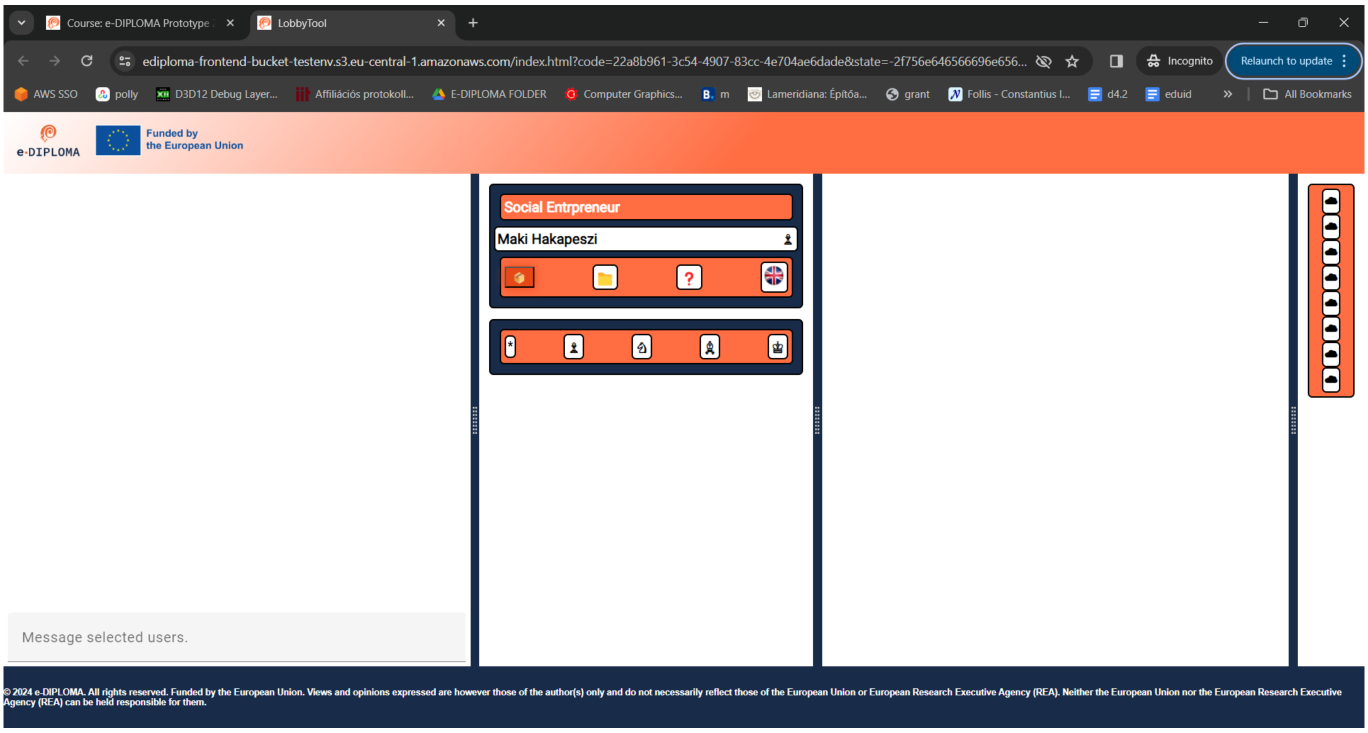


Figure 4:Moodle course page with an e-DIPLOMA activity

When the student opens (clicks on) the activity, they are redirected to the e-DIPLOMA user interface, opening in a new tab of the browser. Which page is displayed may depend on the exact type of activity and in which phase the activity is, but the possibilities are:

- the user is taken to the Asset Management screen – this happens if the activity has been posted for the purpose of allowing access to asset managements, without an associated e-DIPLOMA lobby functionality. See section 5.3 for details.
- the user is taken to the e-DIPLOMA Lobby screen. If no Learning Object has been assigned to the Lobby, then a message instructing the student to try later is displayed. If the activity has not been prepared by the supervisor yet (no virtual rooms created, no cloud capacity allocated), users can join the lobby, interact with other users in the lobby via chat, and have prerequisite software and assets downloaded on their device, to save time during actual class. If the supervisor has prepared the lobby, users can join rooms, and, upon supervisor approval, join VR, desktop, or web-based single or multiplayer sessions.

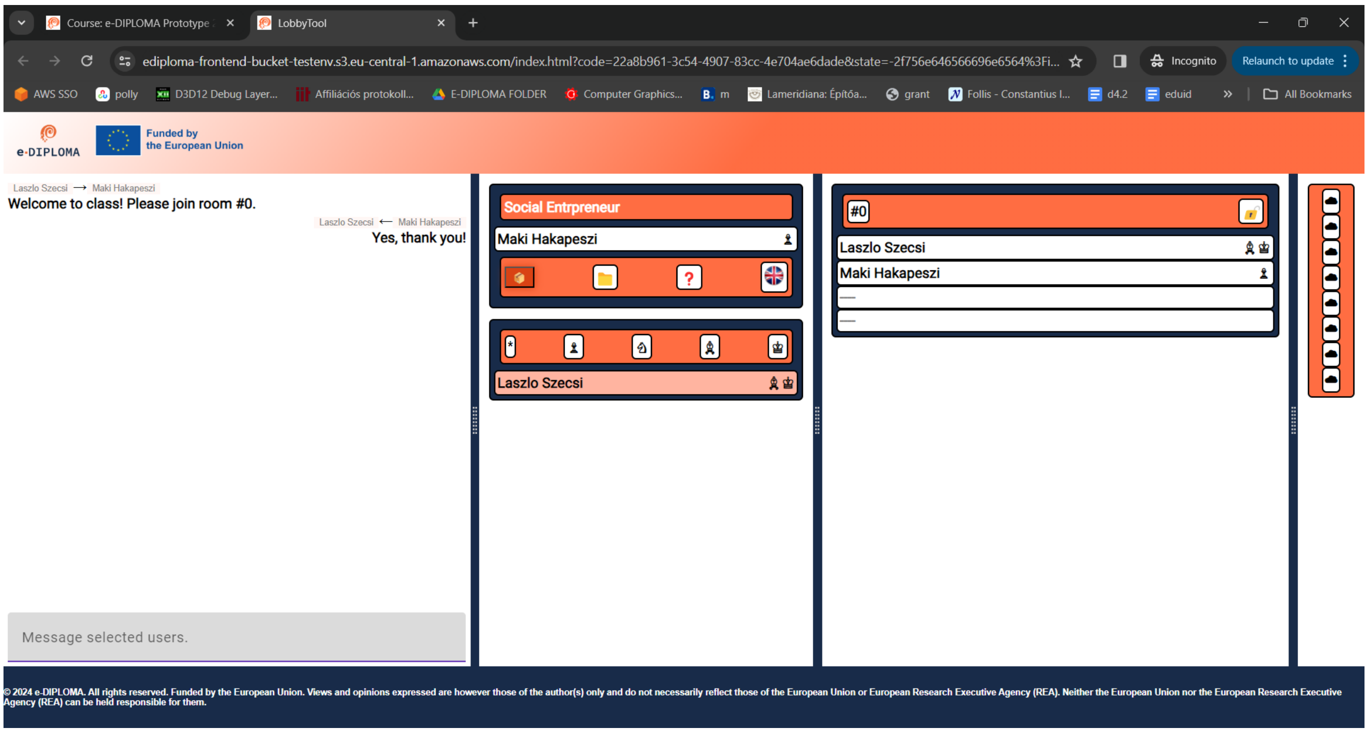
Before opening the Lobby Page, authentication is performed as described in section 8.1. This may take several seconds, during which a white page is shown. An empty Lobby Page appears first.



© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 5: Empty Lobby screen

Then, the lobby is populated by the participants of other online users, and virtual rooms (below).



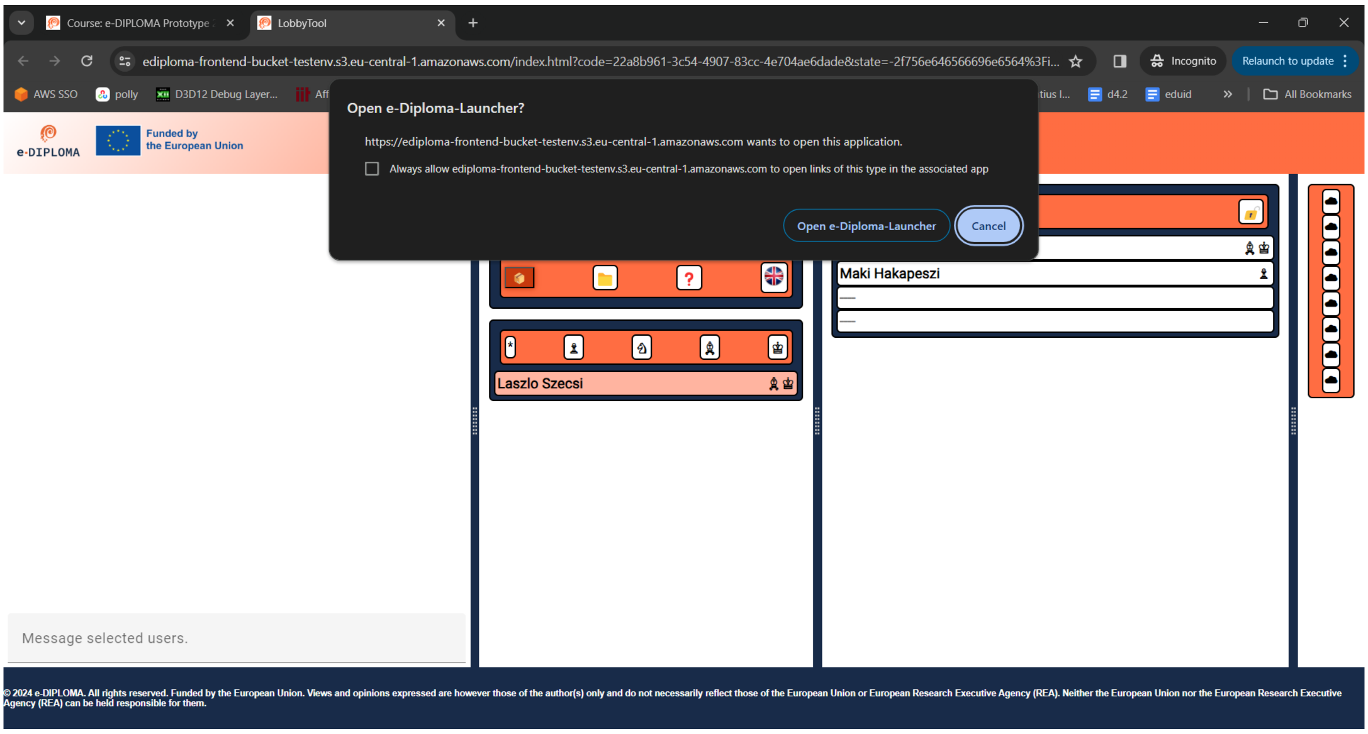
Laszlo Szecsi ⟶ Maki Hakapeszi
Welcome to class! Please join room #0.

Laszlo Szecsi ⟵ Maki Hakapeszi
Yes, thank you!

© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 6: Lobby populated by online users and virtual rooms.

The lobby immediately tries to start the e-DIPLOMA Launcher. If the Launcher has already been installed, the user will be prompted to allow it to open, unless they have previously opted to let the browser start the Launcher silently.


Open e-Diploma-Launcher?

https://ediploma-frontend-bucket-testenv.s3.eu-central-1.amazonaws.com wants to open this application.

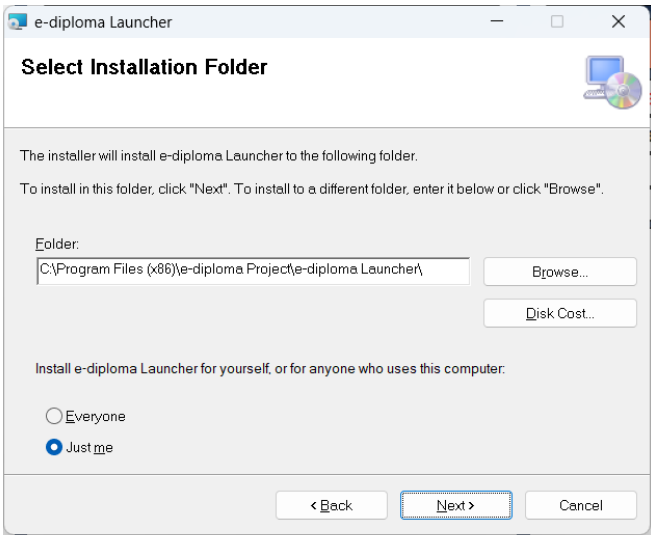
☐ Always allow ediploma-frontend-bucket-testenv.s3.eu-central-1.amazonaws.com to open links of this type in the associated app

Course: e-DIPLOMA Prototype
LobbyTool
ediploma-frontend-bucket-testenv.s3.eu-central-1.amazonaws.com/index.html?code=22a8b961-3c54-4907-83cc-4e704ae6dade&state=-2f756e646566696e6564%3Fi...
Incognito
Relaunch to update
AWS SSO
polly
D3D12 Debug Layer...
d4.2
eduid
All Bookmarks
Open e-Diploma-Launcher
Cancel
Maki Hakapeszi
Laszlo Szecsi
Message selected users.
© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 7: Prompt about starting the Launcher.

The user should click *Open e-Diploma-Launcher*, and preferably check the *Always allow* box to avoid these prompts later. The Launcher prompts about any missing prerequisites (application versions of assets) for the Lobby sessions, and downloads them to the user's computer.

If the Launcher has not been installed, a button to download the installer will be displayed (see 5.4). The button links to the e-DIPLOMA Launcher installer, which is an MSI file. Installation requires administrator privileges on the computer. Opening the file will first display a splash screen asking for confirmation that the program should be installed. Then, the setup wizard asks for the installation folder (see Figure 8). On institutional computers, it is recommended to choose a folder which can be written without administrator privileges (i.e. not *Program Files*). This way students without such rights will also be able to use the platform. The installer registers the launcher in such a way that the lobby can start it when necessary (see section 8.5 for technical details).


e-diploma Launcher

Select Installation Folder

The installer will install e-diploma Launcher to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files (x86)\e-diploma Project\e-diploma Launcher\

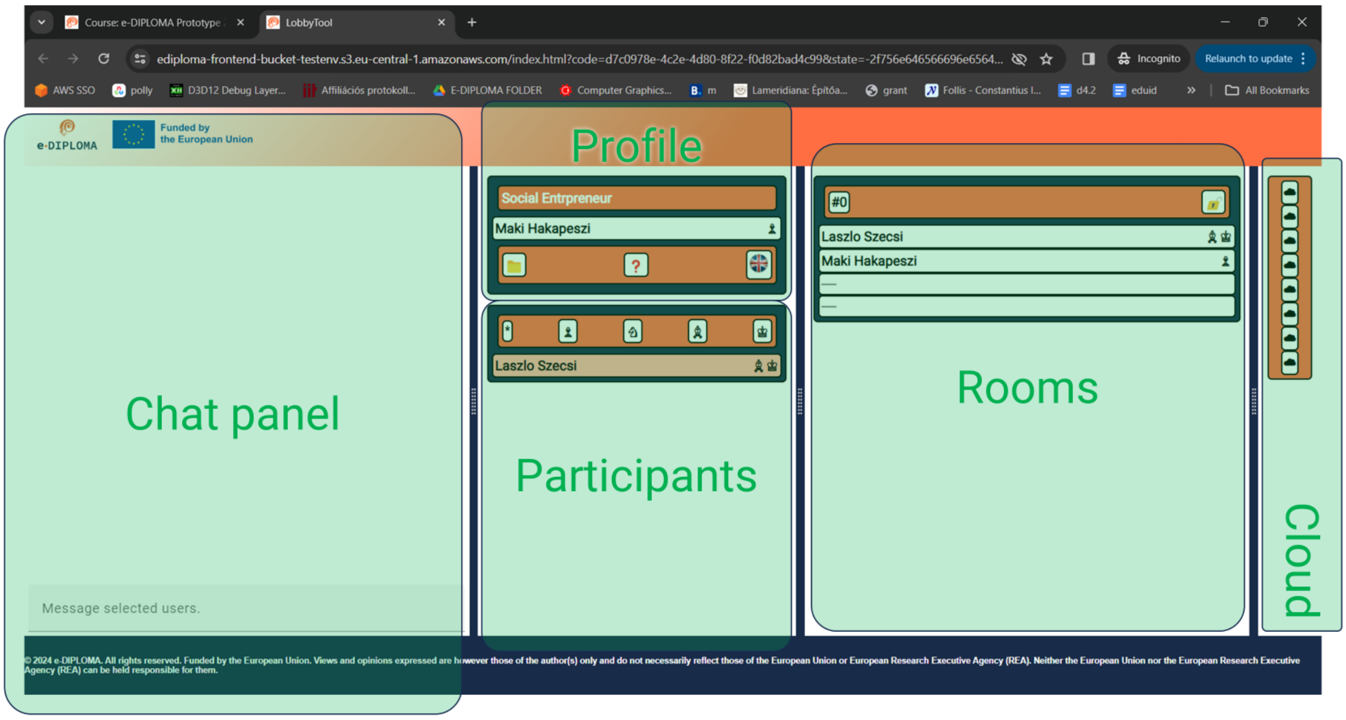
Install e-diploma Launcher for yourself, or for anyone who uses this computer:

Everyone [✓ Just me]

Figure 8: Launcher installer.

The lobby page consists of the following panels (indicated in Figure 9), separated by adjustable splitters:

- the Chat panel
- the Profile and Participants panel
  - the Profile panel is where the Learning Object name, user card, and actions pertaining to the current user are shown
  - the Participants panel is where all other logged in user are shown, and a subset of the available operations like sending chat messages
- the Rooms panel with possibly multiple virtual rooms
- the Cloud resources panel, indicating how many sessions worth of server capacity is available


Course: e-DIPLOMA Prototype
LobbyTool
ediploma-frontend-bucket-testenv.s3.eu-central-1.amazonaws.com/index.html?code=d7c0978e-4c2e-4d80-8f22-f0d82bad4c99&state=-2f756e646566696e6564...
Incognito
Relaunch to update
e-DIPLOMA
Funded by the European Union
Profile
Social Entrpreneur
Maki Hakapeszi
Laszlo Szecsi
Participants
#0
Laszlo Szecsi
Maki Hakapeszi
Rooms
Cloud
Chat panel
Message selected users.

© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 9: Panels of the Lobby Page

5.4.1 The Profile panel

The Profile panel shows the title of the learning object associated with the Lobby page, the card of the user logged in through the browser, and buttons with actions and options that pertain to the user. A user card always displays the full name of the user on the left side, and the icons indicating their roles on the right side. Other features of user cards are only relevant in the Participants and Rooms panels, and will be discussed later. Nevertheless, the user card in the Profile panel can be dragged using the mouse to other panels, e.g. to one of the open rooms to join the room. The profile panel may contain buttons with the following icons:

- 📦: Download Launcher. This button is displayed if the Launcher has not been detected as installed. It is flashing in red to indicate that a crucial task has to be performed before the Lobby would be really functional.
- 📁: Bring assets to room. This button allows the user to select one of their *asset groups*. The assets referenced by this group will be both copied to the game server and downloaded for the game client. Thus, students can display or work with their own projects or files in the virtual environments.
- ❓: Help. This displays the manual page explaining the use of the Lobby.
- 🇨🇿: Language selector. Clicking this button cycles through the available languages. While there is minimal text on the lobby UI, localized tooltips are displayed for every element. At the time of writing this document, not all European languages are supported, and not all texts have been

translated, but a few examples have been included for the demonstration of internationalization of the platform.

- ⏵: Join session. This button only appears if the user has joined a virtual room, a supervisor has approved that room by locking it, initiated the session, and the application server has been started in the cloud. Then, by pressing this button, the user can request the Launcher to launch the client with the appropriate parameters to join the server in the cloud. Then the interactive multiplayer activity can start.

Not all applications on the platform are client-server multiplayer solutions. *Table 2: Participation modalities* lists the possibilities. Notably, the VR technology demonstration activity (see section 7.2) belongs in the Single Player/Client App category, while Brainstorm's Edison technology demonstration (see section Brainstorm Virtual Production tools) is a Web Client/Multiplayer solution, even if, for best performance, the Nice DCV client is preferred over a literal web client. From the point of view of the student user, the setup process is similar, the difference is what happens when the ⏵ (Join session) button is pressed.

Table 2: Participation modalities

	Single player	Multiplayer
Web client	A web application. Redirects the user to the application web page on launch.	Starts the server in the cloud, but does not require the Launcher to download clients. Users are redirected to the app web page with appropriate connection parameters.
Client app	A standalone application. The Launcher downloads and starts it, but no server in the cloud is necessary.	A full-fledged interactive multiuser experience. The Launcher downloads the client and starts it with connection parameters to the cloud server.

5.4.2 The Participants panel

The participants panel appears below the Profile panel. It contains a list of user cards, and above them some selector buttons. The user cards belong to users logged in to the lobby, or recently disconnected. Disconnected users' names are displayed in red, with a timer in the middle of the card, indicating how long they have been offline. Cards can be selected or not selected. Selected cards are indicated by a slight orange hue. Clicking any card selects the particular user, and unselects all others. Clicking while holding the *Ctrl* button toggles selection status. *Shift*+click adds the user to the selection. The buttons featuring the role icons facilitate selection by role. * stands for *all users*. Clicking a button selects users of the indicated role, unselecting others. *Shift*+clicking adds all such users to the selection, *Ctrl*+clicking removes those users from the selection.

User cards can be dragged and dropped. Dropping a user card within the Participants panel rearranges the cards. Dragging a card to a room invites the user to the given room, but does not move them to the room (in contrast to the case when an instructor is doing this, see section 5.5).

5.4.3 The Chat Panel

The chat panel consists of a chat history view, showing sent or incoming messages, and an input field to enter message text. On pressing *Enter*, the message is sent to selected users (see section 5.4.2). Invitations also appear in the chat box. The chat should be used to organise groupings and populate rooms.

5.4.4 The Rooms Panel

The rooms panel contains a number of rooms with a pre-determined number of slots, as created by the supervisor (see section lobby-supervisor). Rooms may be in the following states, indicated by the icon in the upper right corner:

- 🔓: Open. Users can join the room as long as there are free slots. Users can also leave the room.
- 🔒: Locked. This indicates that the supervisor has approved the room setup, and the activity can start. Users cannot join or leave the room.
- ⏳: Waiting. The session has been initiated, and the backend is busy starting the server. Users cannot join or leave the room.
- 🚨: Active. There is a live session with a server running. Users can join the session, or have already joined the session. Users cannot join or leave the room.

Users can join a room by clicking an empty user card, by dragging their user card from the Profile panel into a room, or by being added to a room by a supervisor. They can change rooms by dragging their card from one room to another. If they drag the card from the room to the Profile or Participants panels, they can leave the room.

Once the user has joined a room, and a supervisor locked the room and started the session, students can launch their app clients by clicking the ⏵: *Join session* button in the Profile panel (see section Lobby Page (student role)). The e-DIPLOMA Launcher proceeds to download prerequisites, including assets brought along (see section 5.4) by users in the room. It is advisable to open the activities before the class, to download the prerequisites in advance, and not delay joining the session with last-minute downloads. From this point on, the e-DIPLOMA App is responsible for interaction with the user. Students should refer to the specific App documentation and supervisor guidance.

5.4.5 The Cloud Panel

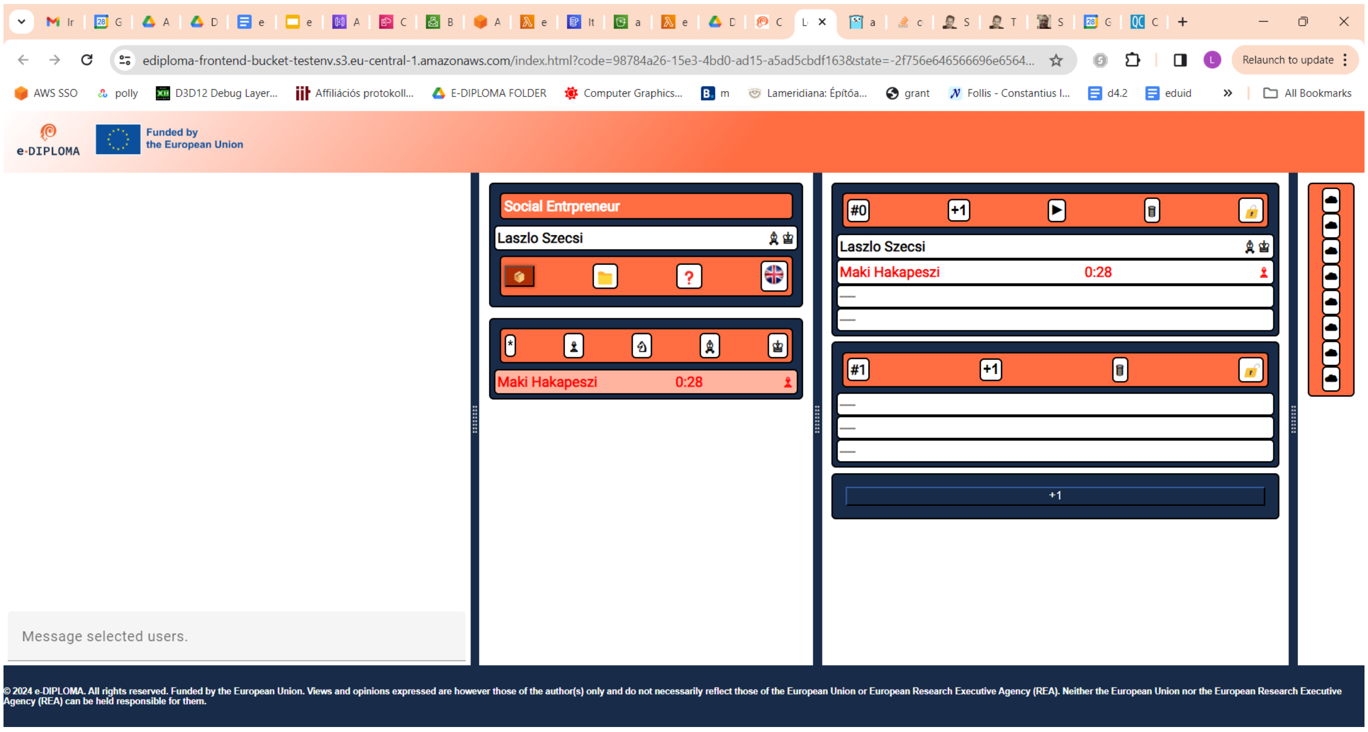
Students do not need to interact with the Cloud panel. However, it is indicated here how much cloud capacity is available, and which rooms are occupying the cloud resources.

5.5 Lobby page (supervisor role)

A supervisor is a teacher who is responsible for presenting a learning object to students in a course, and supervising their activities in the e-DIPLOMA Lobby and in e-DIPLOMA Apps. The corresponding Moodle role is *Non-editing teacher*. However, a Moodle *Teacher*, who is the *Course Manager* in e-DIPLOMA, also has the same capabilities in the Lobby as the *supervisor*. It is the supervisor's responsibility to start the e-DIPLOMA Lobby and ensure the allocation of sufficient cloud resources before the activity starts. Using cloud resources for e-DIPLOMA App servers incurs costs. Therefore, their use should be initiated right before they are needed, and relinquished as soon as they are no longer in use.

The supervisor can open the e-DIPLOMA activity created in a course. If no e-DIPLOMA Lobby exists for the activity, there are two possibilities:

- If the e-DIPLOMA activity settings (custom parameters) specify a learning object, the lobby is automatically created.
- If the e-DIPLOMA activity settings (custom parameters) do not specify a learning object, or the learning object is incorrectly specified and cannot be identified, the Learning Object Selection screen appears. If the course can be identified, the list of learning objects that belong to the course are listed. Otherwise, all courses visible to the user, with all learning objects are listed. Exactly one learning object has to be selected, and then the lobby can be created by clicking on a button.



© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 10: Supervisor view of a Lobby. Note the buttons available for managing the number and size of rooms, opening, locking, starting, and deleting them.

If the lobby for the activity already exists, the supervisor is taken to the e-DIPLOMA Lobby screen. All the options available to students are also available to the supervisor, but they can also move students around the rooms by dragging their user cards. Supervisors can create rooms by using the **+1** *Add room* button under the rooms in the Rooms panel. They can adjust the maximum number of users in a room by using the **+1** *Extend room by one place* button in the top header of a room. Empty user cards can be dragged out of the room to reduce the number of places in a room by one. Supervisors can delete rooms using the 🗑 *Delete room* button. With these tools, the desired number and size of groups working together in virtual environments can be configured with a few clicks.

Supervisors can open and close rooms by clicking the room status icon in the upper right corner of room panels. A locked room can be started by clicking the ▶ *Start room* button. Most importantly, before rooms could be started, appropriate cloud capacity has to be prepared. This capacity is measured in the number of concurrent sessions, and every room started takes up such a session. Once this capacity has been prepared, the costs associated with maintaining the cloud resources apply. These costs can depend on momentary factors, and also whether cloud or on-premise servers are used. After the activity, the supervisor also has to relinquish this capacity. These are done with the ☁ *Prepare cloud capacity* and the 🖥 *Relinquish cloud capacity* buttons in the Cloud panel. The number of possible sessions to be started is indicated by the number of 🖥 buttons. Active sessions occupy a cloud server, and the corresponding cloud capacity button shows the room ID. If ☁ *Prepare cloud capacity* is clicked, new servers **up to the number of the button** are started. As starting a server can take some time, a timer indicates when will be possible to start a room using that cloud server. The supervisor should allocate adequate cloud capacity right before, or at the beginning of the class. All capacity is automatically relinquished at the end time of the activity as specified in the LMS as a custom activity parameter (see section 5.6), or four hours after the start of the activity, whichever comes first. It is most economical, however, for the supervisor to relinquish all cloud capacity as soon as it is not needed. See section 6.3.1Cloud Costs for the typical costs of cloud resources.

5.6 Course management in Moodle

A course manager is a person who creates activities in the LMS. The corresponding Moodle role is *Teacher*. Often, this is the same person as the supervisor who oversees the activity, but not necessarily.

In Moodle, the e-DIPLOMA course administrator is a *teacher* who can edit the course page. In this section, we first describe how to create an activity that launches the e-DIPLOMA Lobby for a given learning object. Later in this section, we discuss how activities that lead to other e-DIPLOMA user interface pages can be created.

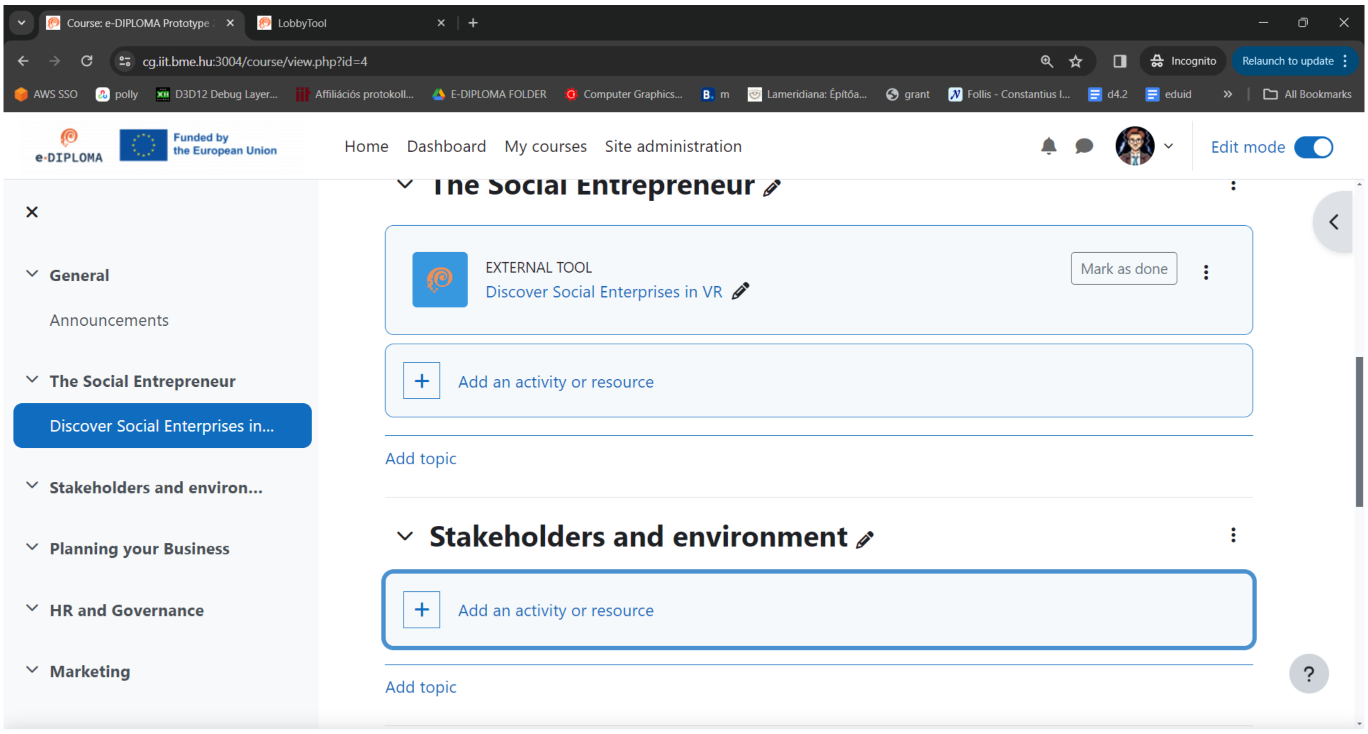


Figure 11: Moodle course page in edit mode.

Funded by the European Union

For adding activities to the course page, *edit mode* must be turned on. In every section, there is an "Add an activity or resource" button. e-DIPLOMA activities are available in the list that appears, distinguished by the e-DIPLOMA logo (if set up by the Moodle administrator, see section 5.7).

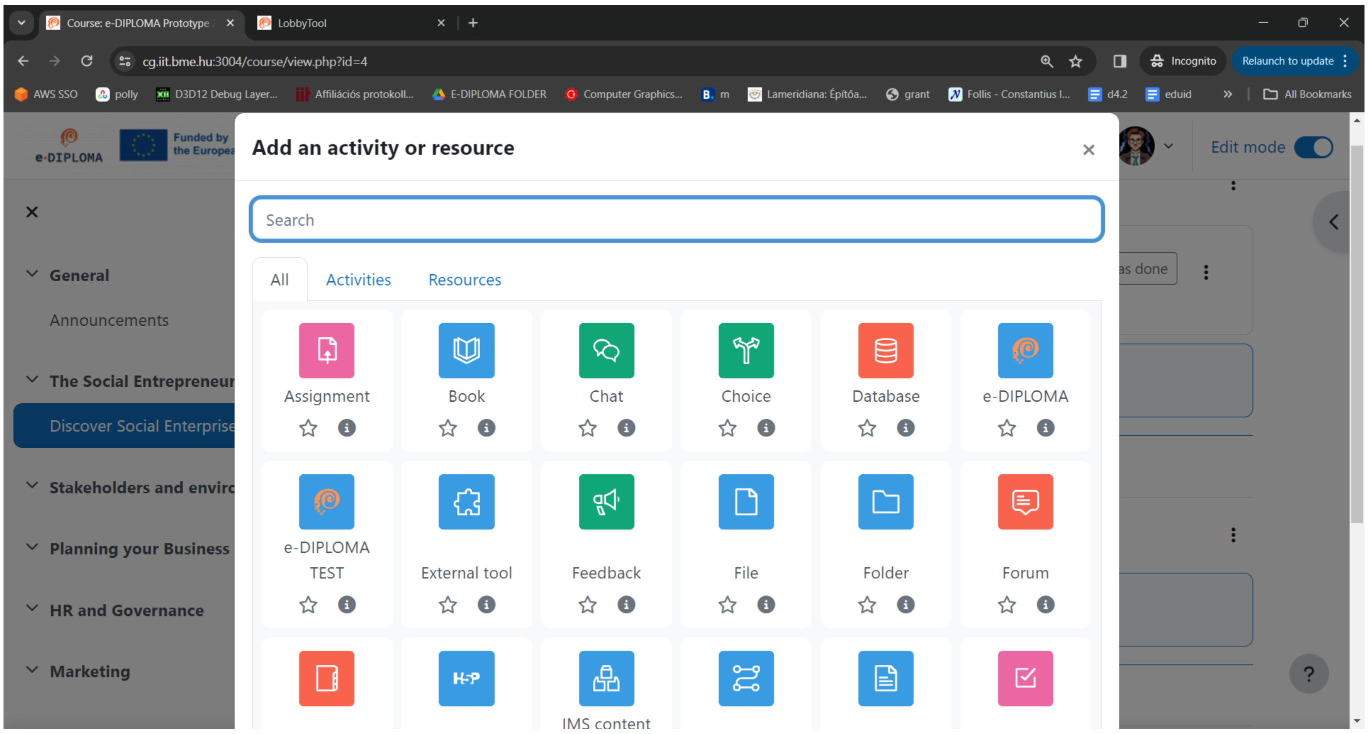


Figure 12: The activity selection screen for Add activity in Moodle. The e-DIPLOMA external tool activity is listed.

Once the activity type is selected, Moodle displays its edit "Edit settings" page. At the minimum, a name for the activity has to be specified. For other settings, you may refer to Moodle's documentation. Under the link "Show more…", the "Custom parameters setting" can be accessed. Here, the text

page=lobby

learning_object=**learningObjectId**

stopCloudTime=**2024-12-19T16:39:57+01:00**

can be entered, replacing learningObjectId with the appropriate identifier. The shutdown time for the preprogrammed release of cloud resources must be given according to the **ISO 8601** standard format [https://datatracker.ietf.org/doc/html/rfc3339], meaning that UTC time has to be used, with an appropriate offset when desired. In the example, the +01:00 indicates UTC+1, or CET, Central European Time. Central European Summer Time (CEST) would be +02:00. The *stopCloudTime* setting can be omitted, but it is a recommended safety feature against unintended cost overruns.

The *learning_object* parameter can be copied from the e-DIPLOMA Learning Object Management page, discussed in the Learning Object section 5.8, but it is also possible to create the activity without specifying the learning object — in this case, the activity should be opened and the learning object can be selected on the appearing Learning Object Selection screen. This can be done by the Course administrator after activity creation, or by the activity supervisor before creating the e-DIPLOMA Lobby for the event. However, as long as no learning object is assigned to the activity, students cannot log in and collect prerequisite applications and assets using the Launcher.

In order to access e-DIPLOMA's management pages, including the *Application Management Page*, the *Learning Object Management Page*, the *Asset Management Page* and the *Consent Management Page* without opening the Lobby, it is possible to create activities that take the user directly to these pages. The *Asset Management Page* and the *Consent Management Page* should be available to all users, so it makes sense to post activities reaching them in every course. The Application Management Page is only relevant for application developers and deployers, and the Learning Object Management Page for creators of learning material. These activities can be placed in a separate course dedicated to e-DIPLOMA management activities.

In order to create a management activity, the same process should be followed as when creating a Lobby activity. However, the custom parameter should be set differently, as given in the following table:

Table 3: Parameters configuration

Application Management Page	page=apps
Learning Object Management Page	page=courses
Asset Management Page	page=assets
Consent Management Page	page=consentForms

Note that on the e-DIPLOMA Moodle server operated for the purposes of the projects, these activities have been created in the course *e-DIPLOMA Management*.

5.7 Moodle administration

The e-DIPLOMA consortium, and specifically BME as work package leader for WP4, operates a Moodle server at https://cg.iit.bme.hu:3004. This server meets the needs of development and the administration of pilot courses throughout the e-DIPLOMA project, and possibly later, if e-DIPLOMA partners continue offering courses independently, outside of adapting educational institutes. However, the e-DIPLOMA platform will be open to educational institutes to facilitate the use of new technologies in their new or existing courses. These institutes would want to use their existing learning management systems. The e-DIPLOMA platform has been designed to work with Moodle, but in theory, any LTI-compliant LMS could be used. In practice, this depends on the idiosyncrasies of the individual LTI implementations, and therefore could require extensive configuration and adaptation, not only on the LMS side, but also in Cloud Administration (see section 6.3). In this section, we focus on how the institute's Moodle administrator can configure their Moodle installation to work together with the e-DIPLOMA platform.

The Moodle version of the eDIPLOMA Moodle server is 4.2.1. LTI capability should be enabled. For better LTI compliance, the Moodle patch #75066, titled *Ability to pass through specific LTI roles to tool provider*, available at https://tracker.moodle.org/browse/MDL-72066, has been applied. This patch is under review at the moment, and it might cause problems with existing LTI tools (not e-DIPLOMA), if they somehow relied on the previous, non-compliant behaviour of Moodle. Not installing this patch limits the ability of e-DIPLOMA to take over Moodle roles. Specifically, the *Non-editing teacher* Moodle role is not passed on as the LTI role *Teaching Assistant*, which means e-DIPLOMA does not recognize these users as *Supervisors* in Lobby sessions. The regular *Teacher* Moodle role is still recognized as a *Course Manager*, with much the same capabilities in the Lobby as a *Supervisor*.

Once Moodle has been set up, one or more users with the system-wide *Administrator* role will exist. These users can also join Lobby sessions, where they are shown as *Administrators*. It is the responsibility of one of these users to set up the Moodle LTI Provider configuration to match e-DIPLOMA. This can be done in the *Site administration* panel, under *Plugins/External tools/Manage tools*. Here, a new tool has to be added, by choosing to *configure a tool manually*. The settings entered depend on the deployment of the e-DIPLOMA platform into the AWS cloud (see section 6.3 for details). In the following we give the settings for the AWS deployment done by the consortium for testing purposes, with identifiers that could be changed for another deployment set in green. If the service is hidden behind another URL, then the URLs need to be adapted. These details are always available from the Cloud service deployer (see section 6.4)

Tool name: e-DIPLOMA

Tool URL: https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod

LTI version: LTI 1.3

Public key type: Keyset URL

Public keyset: https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod/jwks.json

Initiate login URL: https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod/login

Redirection URI(s): https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod/launch

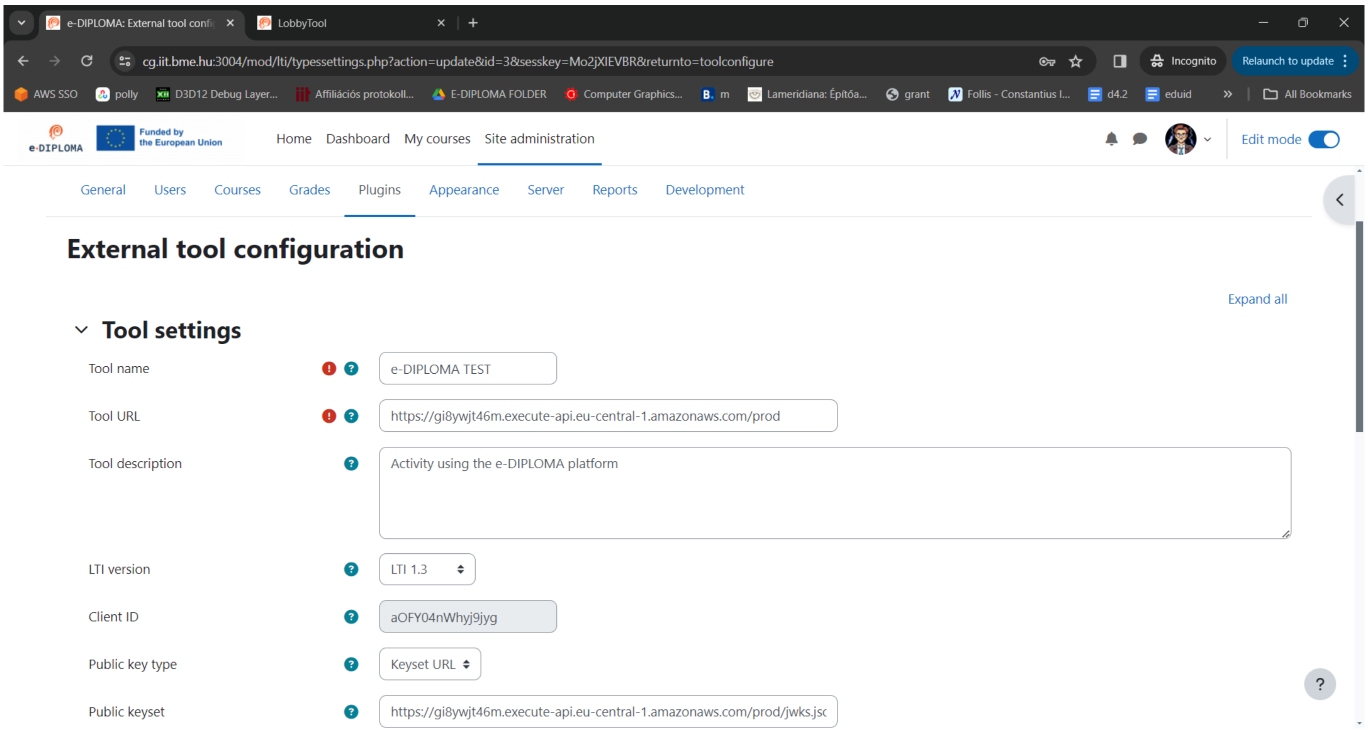
Tool configuration usage: Show in activity chooser and as a preconfigured tool

Default launch container: new window

Supports deep linking: YES

Icon URL: https://ediploma-frontend-bucket-testenv.s3.eu-central-1.amazonaws.com/assets/e-diploma-snail.png

Enable all services.


External tool configuration

Tool settings

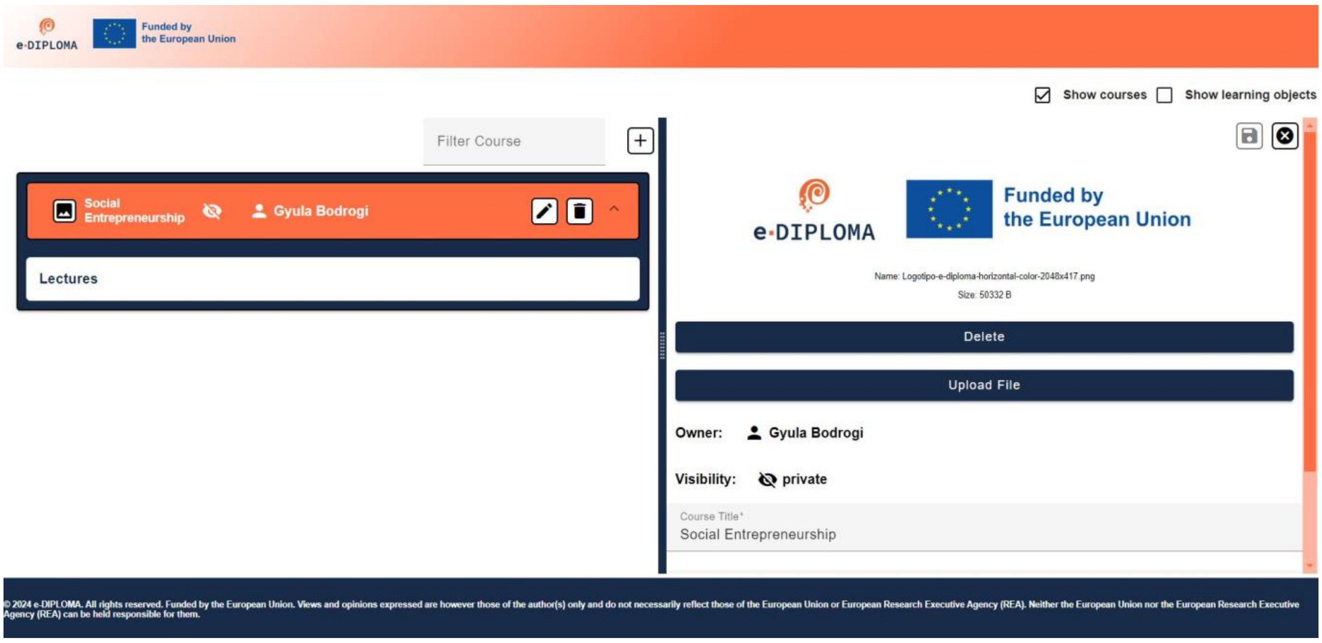
Tool name	e-DIPLOMA TEST
Tool URL	https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod
Tool description	Activity using the e-DIPLOMA platform
LTI version	LTI 1.3
Client ID	aOFY04nWhyj9jyg
Public key type	Keyset URL
Public keyset	https://gi8ywjt46m.execute-api.eu-central-1.amazonaws.com/prod/jwks.jsc

Figure 13: Moodle's External tool configuration page

When the tool has been created, it appears on the same *Site administration/Plugins/External tools/Manage tools* page, under the *Tools* listing. By clicking the three lines, the Tools configuration details can be obtained. These have to be forwarded to the Cloud administrator, so that the LMS can be registered to the e-DIPLOMA platform cloud backend. All e-DIPLOMA functionality will be available through a single Tool. Creating multiple Tool configurations for e-DIPLOMA is possible, but then all configuration have to be added in the cloud backend, and, as different Moodle tool registration effectively work as different Identity Providers, the user logged in through different tools would have different identities in e-DIPLOMA. Therefore, only one tool is to be registered. Customization of instances of that single tool in course pages is possible using custom parameters (see section 5.6).

5.8 Learning Object Management Page

The learning object author takes an e-DIPLOMA App, populates it with educational and media content, creates the necessary configuration or scripting files, thus creating a learning object that can be experienced by students as an activity through the LMS.



Name: Logotipo-e-diploma-horizontal-color-2048x417.png

Size: 50332 B

Owner: Gyula Bodrogi

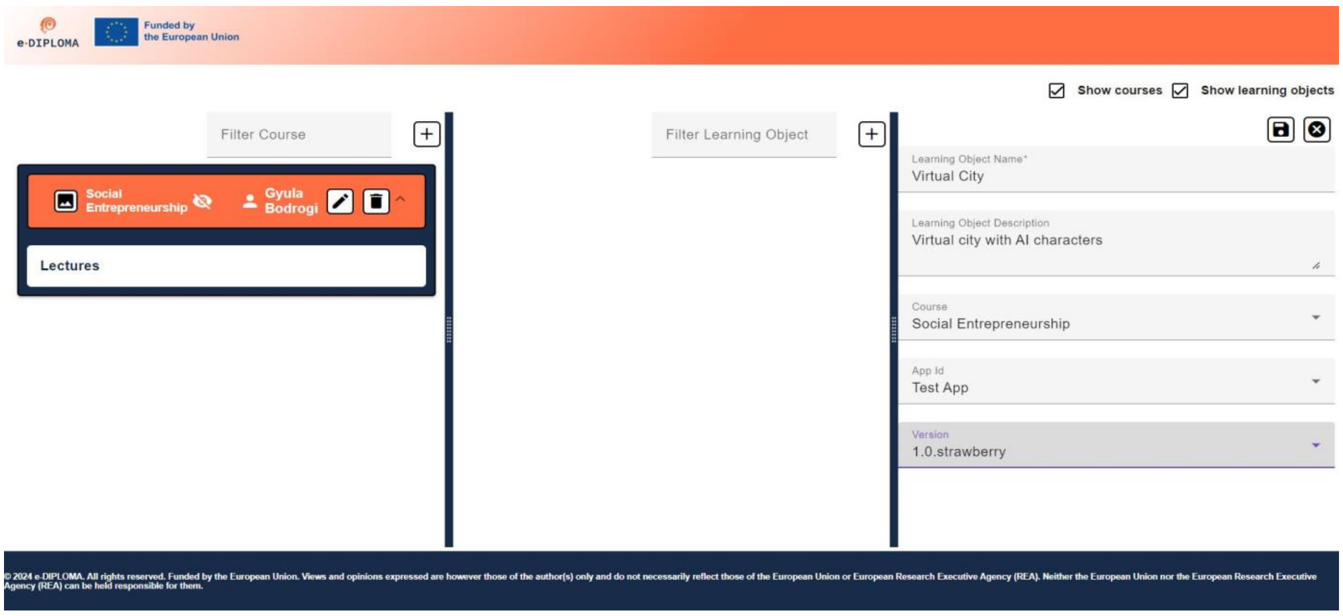
Visibility: private

Course Title*: Social Entrepreneurship

© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 14: Course Details view of the Learning Object Management page

The Learning Object Management page of the e-DIPLOMA platform can be accessed through a custom activity. The page displays learning objects organised by courses. They can be edited, or a new learning object can be added. Every learning object must have an e-DIPLOMA App Build specified, and possibly a client asset group and a server asset group. If the e-DIPLOMA App is single player, and uses no server, then the server asset group is not necessary. If the e-DIPLOMA App uses thin clients (e.g. browser based) that do not require setup, then the client asset group is not necessary. It is unlikely that neither asset group is specified, as that would mean only the assets baked into the e-DIPLOMA App are going to be used. A full client-server app would require both client and server asset groups. Learning objects must be given a title, and they belong to a single course. They are identified by the course and learning object titles.



Learning Object Name*: Virtual City

Learning Object Description: Virtual city with AI characters

Course: Social Entrepreneurship

App Id: Test App

Version: 1.0.strawberry

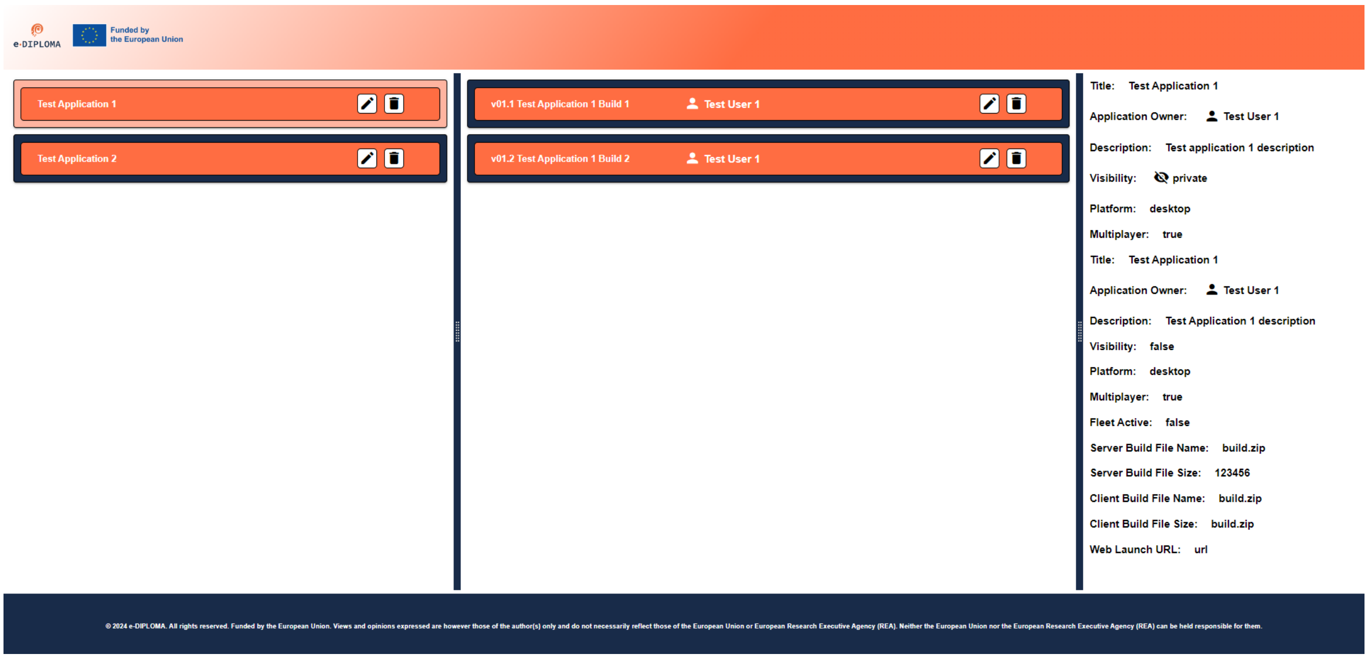
© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 15: Learning Object Details view of the Learning Object Management page

5.9 Application Management Page

An application deployer is part of an expert group that develops an e-DIPLOMA App, and is responsible for testing and releasing the Builds of the App through the platform. A Build is a certain version of the App. Learning Objects are always authored based on a specific version. While it is possible to change a Build, that should only be used to patch errors without changing the interface or functionality, thus possibly breaking existing learning objects. New functionality should always be released as a new Build. Builds are identified using the application title and the build's version number.

A build may consist of a client and a server archive. Single user Apps may only have client archives, thin client Apps may only have server archives. An archive is a collection of files needed to execute the application client on user computers, and the server on cloud or on-premise machines. These files typically include binary executables. Core media assets may be included, but learning object assets and user files are not part of the Build — they are made accessible on the machines after the Build has been deployed to them.



Test Application 1

Test Application 2

v01.1 Test Application 1 Build 1 — Test User 1

v01.2 Test Application 1 Build 2 — Test User 1

Title: Test Application 1
Application Owner: Test User 1
Description: Test application 1 description
Visibility: private
Platform: desktop
Multiplayer: true
Title: Test Application 1
Application Owner: Test User 1
Description: Test Application 1 description
Visibility: false
Platform: desktop
Multiplayer: true
Fleet Active: false
Server Build File Name: build.zip
Server Build File Size: 123456
Client Build File Name: build.zip
Client Build File Size: build.zip
Web Launch URL: url

© 2024 e-DIPLOMA. All rights reserved. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency (REA). Neither the European Union nor the European Research Executive Agency (REA) can be held responsible for them.

Figure 16: Application Management page.

The e-DIPLOMA App Management page (see Figure 16) lists the builds organised by applications. New builds can be added, and existing ones can be patched.

A Build can only be used if it is backed by an AWS GameLift Fleet. As there is a limit on the number of possible fleets for an AWS account, it is possible that not all Builds can have a backing fleet. On the e-DIPLOMA App Management page, Builds can be deployed to fleets, or the fleet of the build may be relinquished. The creation of a fleet is a long process that can take 20-40 minutes.

6 Development

6.1 Application development

An application developer is a programmer, artist, or other kind of game developer who creates new e-DIPLOMA Apps. This is aided by the e-DIPLOMA App Stubs — one for Unity and one for Unreal Engine.

6.1.1 Unreal Engine

The UE Stub uses Unreal Engine 4, because of its robust support of both AWS GameLift and the HP G2 Reverb Omnicept VR headset that the e-DIPLOMA project is using. It is based on the UE4 InitialProject, with basic third-person character control and animation, with multiplayer options enabled. This project is available in the Git repository under the name e-DIPLOMA-UESample-App. Note that this application also serves as one of the prominent Technical Demonstrators of the deliverable, showcasing the multiplayer matchmaking element of the platform (see section 8.1).

The UE Stub uses the AWS SDK to communicate with GameLift and the backend. The repository contains the appropriate build scripts in the *Script* folder, and the required plugins in the *Plugins* folder. The UE Stub calls GameLift SDK methods to send information to GameLift, and it receives events from GameLift in the form of callback method invocations. It notifies the backend through Amazon's SNS service. GameLift also sends notifications to the same topic directly. The backend has a lambda function handling the notifications (see section 8 for details).

Essential functionality for integration with the e-DIPLOMA platform is mainly located in class AGameLiftDemoGameMode, an AGameModeBase subclass. It stores a reference to

- `FGameLiftServerSDKModule`, which provides access to the GameLift SDK.

The following methods are implemented:

- `PreLogin()`
- `Login()`
- `Logout()`
- `BeginPlay()` registers the following event handler callbacks:
    - `OnStartGameSession()`: invoked by GameLift when a game session is created, passing along the game session object containing game properties and other settings. Most importantly, the SNS topic to send messages for the backend, and the session duration are obtained here. Once the game server is ready to receive incoming player connections, it invokes `GameLiftServerAPI.ActivateGameSession()`.
    - `OnTerminate()`: sends a message to the SNS topic indicating that the game session has ended. Calls `GameLiftServerAPI.ProcessEnding()`.
    - `OnHealthCheck()`: calls `GameLiftServerAPI.ProcessReady()` with logging information.
- `PreLogin` calls `GameLiftServerAPI.AcceptPlayerSession()`
- `Login` processes all information from the player, including player name, which can be displayed in-game later. Connecting clients are recorded and managed.
- `Logout` removes a client.

Application developers need to keep this functionality while adding any additional game logic, or even further communication with the backend, e.g. for the purposes of saving activity logs to e-DIPLOMA assets for research analysis, or for submitting grades to be recorded in the LMS.

6.1.2 Unity stub

The Unity Stub uses Photon multiplayer and the GameLift SDK.

Table 4: Methods employed

ProcessReady()	Notifies Amazon GameLift that the server process is ready to host game sessions.
ActivateGameSession()	Notifies Amazon GameLift that the server process has activated a game session and is now ready to receive player connections. Typically called in OnStartGameSession()
AcceptPlayerSession()	Notifies Amazon GameLift that a player with the specified player session ID has connected to the server process and needs validation. Amazon GameLift verifies that the player session ID is valid.
ProcessEnding()	Notifies Amazon GameLift that the server process is terminating.
OnStartGameSession()	Called after the game session has been started.
OnUpdateGameSession()	Called if the game session state has changed. Could be used for player authentication. (Calling AcceptPlayerSession if it succeeded.)
OnProcessTerminate()	Called if the game session has been closed.
OnHealthCheck()	Called periodically with a custom interval. Could be used to check players' state, and kick from session if disconnected.
GameSession	Represents a game session, with a name, fleet id, player count etc.

ServerParameters	Information used to maintain the connection between an Amazon GameLift Anywhere server and the Amazon GameLift service.
Player	Represents a player in matchmaking. When a matchmaking request starts, a player has a player ID, attributes, and possibly latency data.

These must be implemented in order to be able to communicate with the GameLift API. For development purposes there is a CLI implementation of the 12 GameLift API, which runs locally as a small Java application. The callbacks can be triggered using the necessary commands, and it also reacts to the implemented API methods.

6.1.2.1 Plugins

In order for Fusion and GameLift to work, the corresponding libraries and binaries had to be imported. The Fusion library is in the Photon directory in the project 14 root, while the imported GameLift binaries are placed in the Plugins folder. The most important functionality is in class `GameLiftServerManager`. Implemented methods are:

- `Start()`: Entry point, launches GameLift.
- `OnDestroy()`: Exit point, destroys GameLift session.
- `OnStartGameSession()`: Called when GameLift session has been started, starts Photon session, and sets GameLift session status to ACTIVE.
- `OnUpdateGameSession()`: Called when GameLift session has been updated.
- `OnProcessTerminate()`: Called when the GameLift process has been terminated.
- `OnHealthCheck()`: Called periodically, can be used to check game state.
- `GetPort()`: Reads port from argument.
- `EndGameSession()`: Can be used to end GameLift session.
- `NetworkRunnerStart()`: Calls the network runner initializer.
- `InitNetworkRunner()`: Initializes network runner and Photon session.
- `NetworkInputData()`: Represents the input data to be sent over the network.
- `NetworkPlayer()`: Handles network players.
- `Spawner()`: This class serves as a spawner that handles player spawning, input synchronization, and various network-related events within the Fusion. Utils
- `CommandLineUtils()`: Utility class for handling command line input.
- `CommandLineUtilsException()`: Exception made for missing input.
- `UnityMainThreadDispatcher()`: Utility class for dispatching the main thread for Photon.
- `Utils()`: Contains other utility methods, like picking a random coordinate in a circle.

The Unity scene contains multiple prefab objects, which will be spawned when the game starts. Such prefab objects are `GameLiftManager`, `NetworkPlayer`, `NetworkRunnerPrefab` and `UnityMainThreadDispatcher`. Each got the corresponding scripts attached. These all have numerous configuration options in the Unity Editor, most of which are in the Fusion implementation as default, and for which the reader is referred to the Fusion documentation [17].

6.2 Engine integration

Following the pattern of the UE and Unity examples, developers could integrate any game engine into the e-DIPLOMA platform. The process is, however, predicated on the availability of the AWS and GameLift SDK-s in the engine. In the engine used in the stubs, these are integrated as plugins. There exists a GameLift plugin for Unreal Engine 5 since September 2023. The GameLift SDK is available for C++ and C# projects, and should be adaptable to any engine with sufficient effort.

Note that it is possible to sidestep the entire GameLift management process and use EC2 instances directly. This is the path taken with Brainstorm's Edison, which has not been adapted to GameLift (see section 7.1). This, however, needs more manual management of the cloud resources, and may end up slightly more costly when scaling up to more parallel instances needs to be supported.

6.3 Cloud administration

The cloud administrator is an expert who has appropriate access to the AWS account to which e-DIPLOMA has been deployed (see section 6.4 for the deployment process). The cloud administrator should set budgets, warnings, and otherwise monitor the operation and costs of the service. The most important task of the cloud administrator is setting up the connection between Moodle and the platform from the AWS side. This has to be done for every new Moodle External tool configuration. It is recommended that there is only one such registration per institute, thus only one such registration per Moodle connected to the e-DIPLOMA Platform.

The e-DIPLOMA User Pool in AWS Cognito must already exist, and eLTI must be deployed (see section 6.4 for the deployment process). A new OIDC Identity provider has to be added to the user pool. This can be done using the AWS console (Cognito service, user pool configuration page, Sign-in experience tab), but also by running a yaml script in CloudFormation — an approach known as IaC (infrastructure as code).

```
eDiplomaELTIIdentityProvider:
  Type: AWS::Cognito::UserPoolIdentityProvider
  Properties:
    UserPoolId: !Ref eDiplomaUserPool
    ProviderType: "OIDC"
    ProviderName: "eLTI"
    ProviderDetails:
      client_id: "K3xsVRt2SFeFIJP"
      attributes_request_method: "POST"
      oidc_issuer: !Sub ${issuer}
      attributes_url: !Sub https://${eltiApiId}.execute-api.eu-central-1.amazonaws.com/prod/tokenProxy
      authorize_url: !Sub https://${eltiApiId}.execute-api.eu-central-1.amazonaws.com/prod/authorizerProxy
      jwks_uri: !Sub ${issuer}/mod/lti/certs.php
      token_url: !Sub https://${eltiApiId}.execute-api.eu-central-1.amazonaws.com/prod/tokenProxy
      authorize_scopes: "email profile openid"
    AttributeMapping:
      email: "email"
      "custom:ediploma-context":       "https://purl.imsglobal.org/spec/lti/claim/context"
      "custom:ediploma-custom":        "https://purl.imsglobal.org/spec/lti/claim/custom"
      "custom:ediploma-deployment":    "https://purl.imsglobal.org/spec/lti/claim/deployment_id"
      "custom:ediploma-link":    "https://purl.imsglobal.org/spec/lti/claim/resource_link"
      "custom:ediploma-longroles":     "https://purl.imsglobal.org/spec/lti/claim/roles"
      "family_name": "family_name"
      "given_name": "given_name"
      "name":  "name"
      "username":  "sub"
```

In the above script, settings typeset in red come from the Moodle Externel Tool registration (see section 5.7), while settings typeset in blue come from eLTI (see section 6.4). This means that first eLTI has to be deployed, then a Tool registration created in Moodle, and only after that can the identity provider be configured as above. The CloudFormation yaml file is available as part of the e-DIPLOMA-platform-backend code repository as `cloudformation/e-diploma-cognito-stack.yaml`.

6.3.1 Cloud Costs

In the platform specification D4.1, the following costs were foreseen:

GameLift – https://aws.amazon.com/gamelift/pricing/

- FlexMatch: $20.00 per 1 million Player Packages + $1.00 per 1 Matchmaking Hour
- GameLift-managed instances (virtual machines): pay-per-second billing, depends on instance type.
- Data Transfer OUT from Amazon GameLift Instances to Internet: multi-tier pricing (per GB)

DynamoDB – https://aws.amazon.com/dynamodb/pricing/

- Two different pricing models: on-demand vs provisioned.
- On-demand capacity mode: pay-per-request, suitable for unpredictable application traffic.
- Provisioned capacity mode: suitable for predictable application traffic
- 25 GB storage + 25 provisioned Write Capacity Units + 25 provisioned Read Capacity Units per month is free (enough to handle up to 200M requests per month, according to Amazon)

SNS – https://aws.amazon.com/sns/pricing/

- $0.50 per 1M requests, first 1M requests per month is free.
- Email notifications: $2.00 per 100 000 notifications
- No charge for deliveries to Lambda
- Data transfer into SNS: free
- Data transfer from SNS to Lambda: multi-tier pricing (per GB)

EC2 – https://aws.amazon.com/ec2/pricing/

- Virtual Machines: pay-per-second billing, depends on VM configuration and region.
- Data transfer from Internet to Amazon EC2: free
- Data transfer from Amazon EC2 to Internet: multi-tier pricing (per GB)
- Pricing for volume storage (Amazon EBS): https://aws.amazon.com/ebs/pricing/

Lambda – https://aws.amazon.com/lambda/pricing/
- $0.20 per 1M requests, first 1M requests per month is free.
- Compute time × memory usage (GB-seconds), multi-tier pricing, cost depends on selected CPU architecture (x86 or Arm) and memory configuration, first 400 000 GB-seconds per month is free.

S3 – https://aws.amazon.com/s3/pricing/
- Object storage (per GB-month) depends on storage class.
- Number of requests, depends on request type.
- Data transferred from an Amazon S3 bucket to any AWS services within the same AWS region as the S3 bucket.
- Data transferred from Internet to Amazon S3 is free.

API Gateway – https://aws.amazon.com/api-gateway/pricing/
- WebSocket connections: $0.25 per million connection minutes
- WebSocket message transfers: $1.00 per 1M (first 1 billion), $0.80 per 1M (over 1 billion)
- Data transfer: same as EC2 data transfer rate

During the tests conducted for the development of the e-DIPLOMA platform, only the cost of EC2 instances and the GameLift service were above the limits of the free tier of Amazon. We use g5.large instances, with a daily cost up to 7.00$. This is for a single server for 24 hours. As an example, using 5 servers for 5 groups of 4 students for a two-hour class would cost approximately 3$. However, releasing the cloud resources after the activity is crucial, therefore multiple safeguards have been implemented: game server session time out after a set amount of time, cloud capacity allocated in the Lobby are released after 4 hours, and a time to release them earlier can be set as a custom parameter of e-DIPLOMA activities in Moodle.

6.4 AWS deployment

The entire e-DIPLOMA platform has been deployed to AWS. It has been configured to be accessible from the e-DIPLOMA Moodle server. However, the capability to re-deploy to other accounts is indispensable, if later one of the partners takes over the operation of the cloud infrastructure from BME, or if other entities, maybe even educational institutions, would operate their own services instead of joining their LMS systems to the existing backend. For that reason, the entire platform has been developed as an Infrastructure-as-code solution, using AWS Cloudformation and AWS CLI. Cloudformation uses yaml scripts to automate the creation of *stacks*, i.e. collections of AWS resources. CLI stands for command line interface, meaning individual commands can be issued to perform operations that could otherwise be accessible from the AWS Console.

The e-DIPLOMA IaC deployment relies on the following components:
- eLTI deployment
- CloudFormation templates
    - e-DIPLOMA-platform-backend/cloudformation/e-diploma-platform-backend.yaml
    - e-DIPLOMA-platform-backend/cloudformation/e-diploma-cognito-stack.yaml
- AWS Lambda functions written in Golang
- build and deployment scripts for the Lambdas
    - e-DIPLOMA-platform-backend/scripts/build-lambda-golang.bat
    - e-DIPLOMA-platform-backend/scripts/deploy-lambda-golang.bat
- Angular Typescript source code for the Lobby and Management frontends
- build and deployment scripts for the frontend
    - e-DIPLOMA-Lobby/deploy.bat

7 Technical demonstrator applications

7.1 Brainstorm Virtual Production tools

The content of the class of the future, where the professor could be embedded in virtual or augmented reality scenarios (Figure 17), projecting animated 3d assets and all kinds of multimedia content as like as visualise this content trough the most advanced visualisation devices (i.e. VR headsets) would be possible just employing a virtual production authoring tool. In the e-Diploma Project these tools are provided by the integration of Brainstorm SW, Edison and the project platform. Brainstorm Edison is a software that permits creating presentations exploiting virtual production techniques. The software workflow and interface have been adapted to non-expert users to simplify the adoption in educational settings.

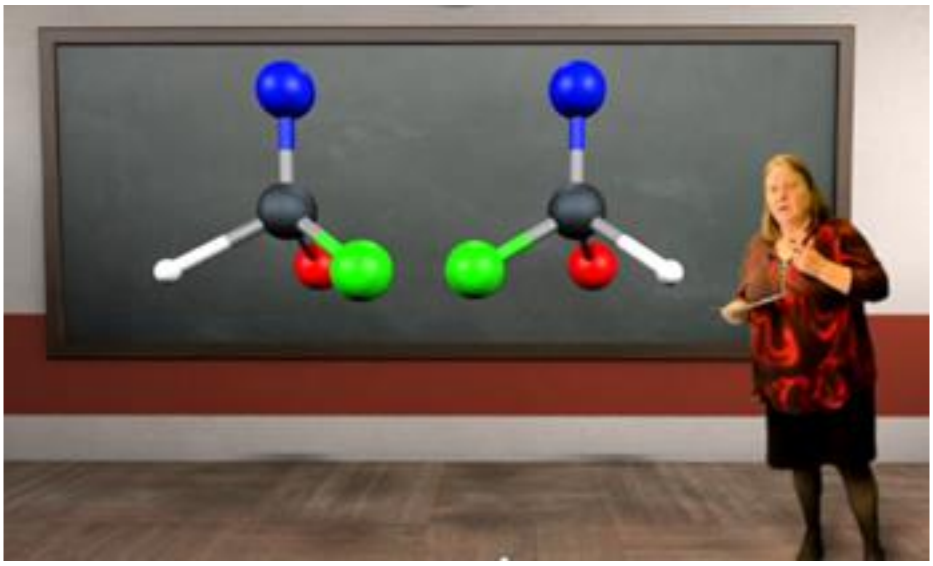


Figure 17: Virtual production based educational content example.

One of the main aspects that tackle the wide adoption of the software, are the hardware requirements. Virtual production tools are highly demanding in terms of graphical processing and consequently requires a powerful computer to run. This point will be faced in the e-Diploma project, virtualizing Edison employing AWS tools. This allows it to make the solution available for the e-Diploma platform and deploy it on demand in order to limit the cost of maintaining such demanding tools, and do it just when it is needed by platform users.

The Virtual production solution for the e-Diploma platform has several embedded tools that allow the educational users to begin editing the virtual production content, starting from the contents they commonly use to carry out their courses. Actually, the most employed tool to carry out educational sessions are presentations, normally edited employing Microsoft Powerpoint. To start using Edison and edit a presentation using this tool, the first step is the selection of the contents to be included in the Edison project, starting from powerpoint (.ptt) or common portable document format (.pdf) files. The first step for the professor, in order to prepare the educational session and the connected Edison project, will be the selection of materials (powerpoint file, graphs, pictures, movies, etc) needed from its files in moodle. Once selected, the professor will start the Edison virtual machine, selecting the tool from the platform control through the web interface. This trigger will instantiate the AWS image saved where Edison and its connected tools are installed.

Once instantiated, the platform backend will copy the files, the professor selected, to the virtual machine, placing them in the Edison resources folder. When the professor accesses (more details in section 4.5), the first visualised is the Quick Start wizard (add image) interface. This Edison tool guides the professor in the preparation of the previously selected assets, to be properly imported in Edison and to mount the slides (in case of Edison: images, 3D models, PPT files, movies, graphs, etc.) sequence. As stated, the most common starting point is a presentation. When a file with these characteristics is selected to be used in Edison, the first process applied is the exportation of the powerpoint slides to Edison format. This is carried out automatically employing a proprietary PowerPoint plugin from Brainstorm. The process is transparent for the user. The powerpoint opens and the slides are then exported to video files (allowing the export of powerpoint slides animations in the generated videos). The generated videos are collected in a folder that is created in the same folder the PPT file is placed. The video slides are then included in the Edison project file. This same process is carried out in case PDF files with multiple pages are selected to be used as slides. Once the starting assets are imported and the project created, the professor can now pass to adjust the content, add the text to be displayed in the teleprompter for every slide, and all the following related tasks to prepare the presentation/content for the educational session.

Once the session's content is ready for its production, being the objective of a recorded session or an interactive live class, the professors will leverage different tools embedded in the Edison virtual machine.

In the case the session would be recorded, the professor will access the Edison VM, and will start the connected tool to record the class.

In the case the class would be live streamed, the professor will start the MS Teams in its local machine and will share the browser window where the Edison VM and the related content is being visualised and managed.

In the case the class would be live streamed and the content requires some interaction with the invited students, the professor will access the MS Teams installation in the Edison VM. In this way the professor will be able to visualise the video feeds from selected students inside the virtual environment and in the case he will provide the access to control the session to some student, the student will be allowed to lead the session and to follow presenting/explaining/managing the content of the session in the same way the professor started.

7.2 VR demonstrator

The virtual reality (VR) learning object demonstrates the capability of the platform to support VR. After connecting a head-mounted display (HMD) and two controllers to a computer, a user can open one of the activities through the platform and have it detect and utilise the HMD and controllers. This was accomplished through the native Oculus and OpenXR support in Unity Engine, together with the XR Interaction Toolkit package available on the Unity Package Manager (UPM). Alternatively, there is a non-VR version of the application, which use an XR Device Simulator plugin to simulate a VR setup without requiring one.

The primary testing application presents an adaptation of an educational and medical illustration style to interactive virtual environments. In future versions, the style will vary throughout the scene, showing the use cases of different styles and helping teachers decide on which style best fits their content.

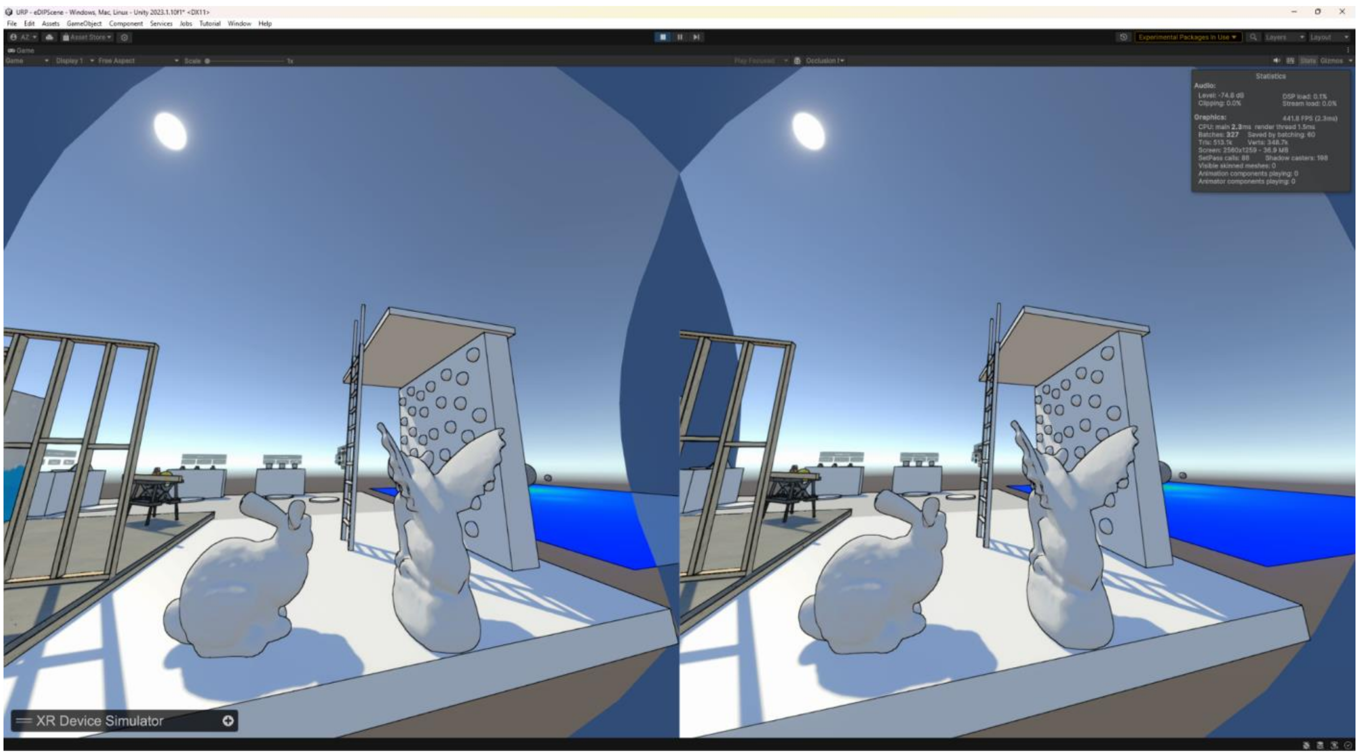


Figure 18: Stereo rendering for VR

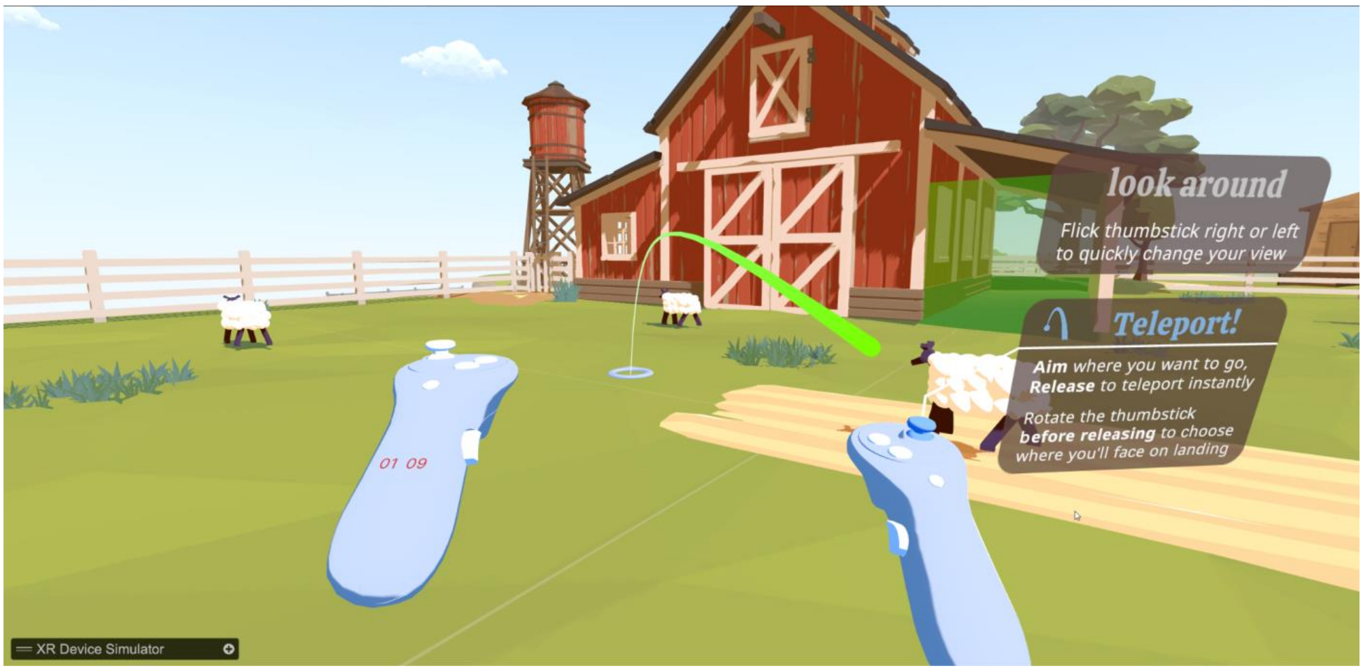

look around
Flick thumbstick right or left to quickly change your view
Teleport!
Aim where you want to go, Release to teleport instantly
Rotate the thumbstick before releasing to choose where you'll face on landing
01 09
XR Device Simulator

Figure 19: VR navigation

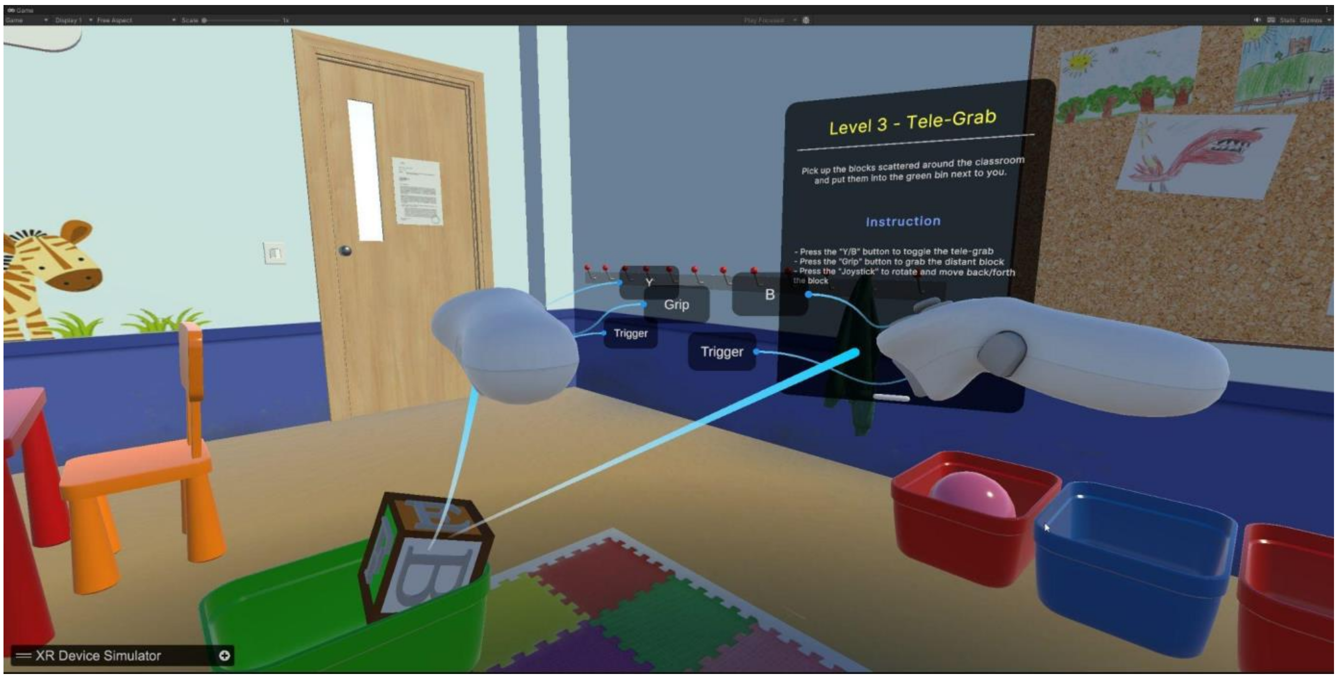

Level 3 - Tele-Grab
Pick up the blocks scattered around the classroom and put them into the green bin next to you.
Instruction
- Press the "Y/B" button to toggle the tele-grab
- Press the "Grip" button to grab the distant block
- Press the "Joystick" to rotate and move back/forth the block
Y
Grip
B
Trigger
Trigger
XR Device Simulator

Figure 20: Object manipulation in VR.

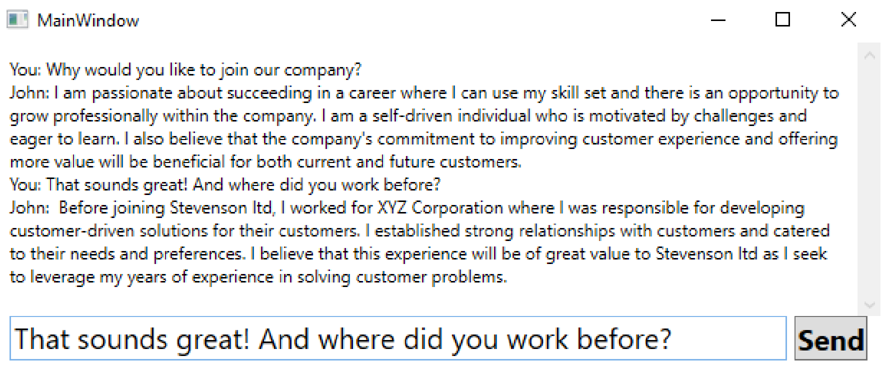
7.3 AI Conversation Demonstrator

This application demonstrates our preliminary capabilities to integrate dynamic and responsive AI personalities into selected games. Our goal is to provide a service that is easy to integrate and enriches the learning experience with lifelike and emergent behaviour. An application can prepare an AI personality by providing a textual description of its behaviour, memories, and goals. Then the player can communicate with it by asking questions in an interview or trying to negotiate a business arrangement.

The service runs on a powerful remote server with dedicated GPU hardware. It provides a stateless RESTful API (OpenAI API) that expects the message history of a given character and provides a continuation of the conversation. The service should be accessed by a game server running on the same network. The server is responsible for maintaining the state of a character by remembering the conversation. The client sends the next user message to the server and the server responds with the last answer of the AI.

Our current implementation uses the publicly available Alpaca-7B language model by WeOpenML, but we are also investigating other models. On the backend we use the VLLM library to load and process the models. Communication is handled by the OpenAI.Net.Client library. The client app is for demonstration purposes and provides a simple interface to chat with a fictional character called John who is trying to apply for a position at the player's company. A learning object can be assembled with an initial configuration that describes the identity of the AI character in JSON format.

e-DIPLOMA
Funded by the European Union


MainWindow

You: Why would you like to join our company?
John: I am passionate about succeeding in a career where I can use my skill set and there is an opportunity to grow professionally within the company. I am a self-driven individual who is motivated by challenges and eager to learn. I also believe that the company's commitment to improving customer experience and offering more value will be beneficial for both current and future customers.
You: That sounds great! And where did you work before?
John: Before joining Stevenson ltd, I worked for XYZ Corporation where I was responsible for developing customer-driven solutions for their customers. I established strong relationships with customers and catered to their needs and preferences. I believe that this experience will be of great value to Stevenson ltd as I seek to leverage my years of experience in solving customer problems.

That sounds great! And where did you work before?

Figure 21: Conversation with the AI.

7.4 Unreal Engine Multiplayer

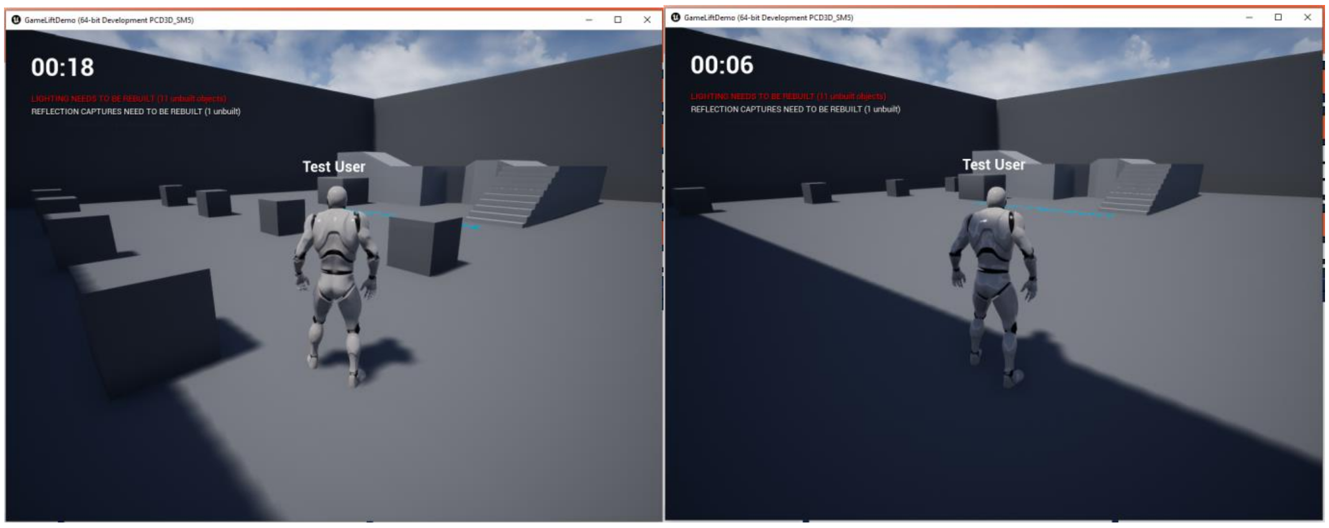

GameLiftDemo (64-bit Development PCD3D_SM5)
00:18
LIGHTING NEEDS TO BE REBUILT (11 unbuilt objects)
REFLECTION CAPTURES NEED TO BE REBUILT (1 unbuilt)
Test User
GameLiftDemo (64-bit Development PCD3D_SM5)
00:06
LIGHTING NEEDS TO BE REBUILT (11 unbuilt objects)
REFLECTION CAPTURES NEED TO BE REBUILT (1 unbuilt)
Test User

Figure 22: Two learning objects using the same application but different assets. Note the difference in the positioning of the boxes in the scene.

7.5 Unity Engine VR Multiplayer

Photon Fusion is a library made for the Unity game engine, and designed to support different multiplayer topologies. It was implemented to make multiplayer game development in Unity easier with more or less integrating it into the usual Unity workflow.

Other solutions have existed for multiplayer networking made for Unity before, like Photon Bolt and Photon PUN. Fusion's development has been going since 2021, aiming to replace and combine these technologies. Bolt and PUN have a target player count around 32 players, which would be enough for our application, but they also lack in some features, like robust tick based simulation, lag compensation or the most important, dedicated server support. These older technologies utilised the shared topology, which means the client can act both as a server as well as a client, and any client can have state authority over some objects. If the game is complicated, this can lead to an unnecessary overhead, it is better to separate them. Fusion is also a more recent, high performance framework, and will have better support in the future.

Host Mode

Host Mode resembles a usual topology, designating one player as the host (server) and others as clients. While it provides a straightforward and compact solution, it lacks the customization flexibility of Server Mode. In Host Mode, the game must encapsulate the code and logic for both roles, potentially leading to confusion and a detrimental impact on performance. Despite these drawbacks, this topology remains optimal for smaller, low-performance games due to its simplicity.

Server Mode

Server Mode typically involves a conventional server-client topology, where a dedicated server handles the game's business logic, manages connecting players, lobbies, and handles player movement and data. Clients, in this setup, have the only responsibility of connecting to the server and controlling the player. This approach offers several advantages over Host Mode. One key benefit is that the server does most of the calculations, so the clients are not required to be high-performant, although this also means the server infrastructure should have strong hardware and be robust to be able to handle several clients at once. Additionally, the server can be independent and can store confident player data more securely. An individual server is also easier to run in the cloud. So in the end, Server Mode gives us even more flexibility and customizability than Host Mode, while performing even better (if server hardware allows it).

Shared Mode

In both Host and Server Mode the designated server has full state authority over all game objects, so when a client changes the game state, first it gets synchronised with the server state, and if needed, the server will overwrite the client's state. In contrast to this behaviour, in Shared Mode the clients can also have state authority over some game objects, more precisely over the game objects they spawn. But the clients are allowed to hand over the authority of any game object to other clients. This mode is similar to the earlier mentioned PUN, meaning it also has its drawbacks. Lag compensation and tick rate syncing is not available, and the game session is live only while all clients are connected. This solution is great for compatibility, but not robust enough for today's standards.

In addition to the Fusion library, Photon also provides a cloud service for handling connections, and also supports relay fallback and state backup for server migration. This enables the client to connect to the server without knowing the exact address possible, it is enough to keep count of the so-called App ID. This way all the connections will be made through Photon in the cloud. On the server side, all we need is a headless build of the game (utilising the Fusion libraries) and a running instance of a Photon Dedicated Game Session running on the dedicated server. Dedicated server-client topology means that only the server has state authority but no input authority, and the clients have only input authority but no state authority. This dedicated server can be hosted on a remote server of a hosting provider, like AWS GameLift. Photon servers are different in this sense, since they only offer to handle the connection, but not the game session itself. They provide a free tier of subscription for development and for limited usage (max. 20 parallel connections), which is more than enough for testing, but in a later phase of production it might be worth subscribing to a higher tier. Managing the game session (server orchestration) is the task of the provider, Fusion has nothing to do with it. We will discuss such a service, AWS GameLift later.

After a Photon account is registered, one can generate Photon Sessions in the cloud from the Dashboard. Creating a session is very simple: in the Dashboard click on "Create a new app", and then choose a name game type (multiplayer), an SDK (Fusion) and a name for the application. 5 Figure 3: Creating a Photon App Next the game session appears with a generated App ID, which has to be passed to the server and the client as well. 2.5 Fusion Framework Basics Fusion introduces two new main components to the library: NetworkRunner and NetworkObject. These elements (classes) are responsible for all the networking and simulation functionality Fusion can offer. The NetworkRunner is our main, core object, which manages both networking and simulation in the scene on the server and on the client as well. NetworkObject is a Unity prefab and can be applied for any scene object which we want to simulate on the network runtime. One can add actual networking and simulation capabilities to game object through behaviours: SimulationBehaviour and NetworkBehaviour. As their names suggest, SimulationBehaviour can be used to add simulation steps and callback, while NetworkBehaviour is used to add networking properties, that comprise the game state, and to hold data which is automatically synchronised across the network.

```
// example of defining a networked variable

[ Networked ] public byte life { get ; set ; }



// example of registering a callback on value change

[ Networked ( OnChanged = " OnTypeChanged ") ]

public Type type { get ; set ; }



public static void OnTypeChanged (

Changed < TheClassWhichHasTheProperty > changed

)

{

// your code here - check API docs for more details

}
```

The [Networked] property can be used in C# style as shown above, and works for all primitives, structs and references to other NetworkObjects, except bool, where [NetworkBool] should be used for optimization reasons. Callbacks can be also defined as shown on change of values, and it is even possible to control where the code should be executed using OnChangedLocal and OnChangedRemote. Fusion offers prebuilt NetworkBehaviours to get a game running quickly, like NetworkTransform, NetworkRigidbody and NetworkCharacterController. These work similarly to built in behaviours, but they keep everything synchronised and smoothly interpolated automatically. Lifecycle Analogue to Unity's Start(), Spawned() is implemented network-safe, it should be generally used where Start() would be used. Awake() still can be used, but only for one-time initialization of things that will not change. There is also FixedUpdateNetwork(), which is called at fixed time intervals, and can be used to execute logic and to

e-DIPLOMA Funded by the European Union

change the network state. Network properties are always reset before calling FixedUpdateNetwork(), and Render() will run after.

Every input is collected into a single struct each tick. This struct can be read inside FixedUpdateNetwork() to control the simulation in the client or the server from a game object with input authority. For single, one-time events the usage of Fusion Input and [Networked] might be not enough, because while they are fast and precise, not very reliable. Packets may get lost, so in the case of critical one-time events one could use Fusion's RPCs (Remote Procedure Calls). There is a fairly simple C# syntax for it. The [Rcp] tag can be added to a function (with a case-insensitive "rcp" prefix or postfix) which returns void. The attribute can be also filtered by where it may be called from or where it gets executed.

```
[ Rpc ( RpcSources . InputAuthority ) ]
   public void RPC_Configure ( string name , Color color )
{
playerName = name ;
playerColor = color ;
}
```

8 Technical documentation

8.1 The e-DIPLOMA Platform Backend

The AWS backend manages Web Lobbies and automatically starts a dedicated server for each virtual room session. A session is a shared virtual experience in which the participants can interact with each other. Sessions can be started with the permission of an authorised person (e.g., a teacher). Dedicated servers are responsible for managing and synchronising the application state between the participants of a given session as well as collecting data about user-specific events, for analytical purposes. By default, the AWS backend launches virtual machines in the cloud, on which the dedicated server is hosted.

AWS Services

The infrastructure behind the e-DIPLOMA backend consists of the following services:

- GameLift – A fully-managed service which simplifies the process of setting up, scaling, and managing dedicated game servers. The service supports a wide range of game engines, including Unreal Engine, Unity, and custom-built engines, and can be used for session-based multiplayer applications. With GameLift, dedicated servers can be deployed to AWS-managed or on-premises infrastructure, as part of a GameLift fleet.
- DynamoDB – A managed NoSQL database service that is designed to be highly scalable and can handle millions of requests with automatic scaling. In the AWS backend, the service is used for storing state information about users and web lobbies, as well as managing an inventory about applications installed to the platform backend hosted by a given institution.
- SNS – A managed messaging service that can be used to publish events and notifications to multiple subscribers or endpoints in real-time, such as email addresses, mobile devices, or AWS

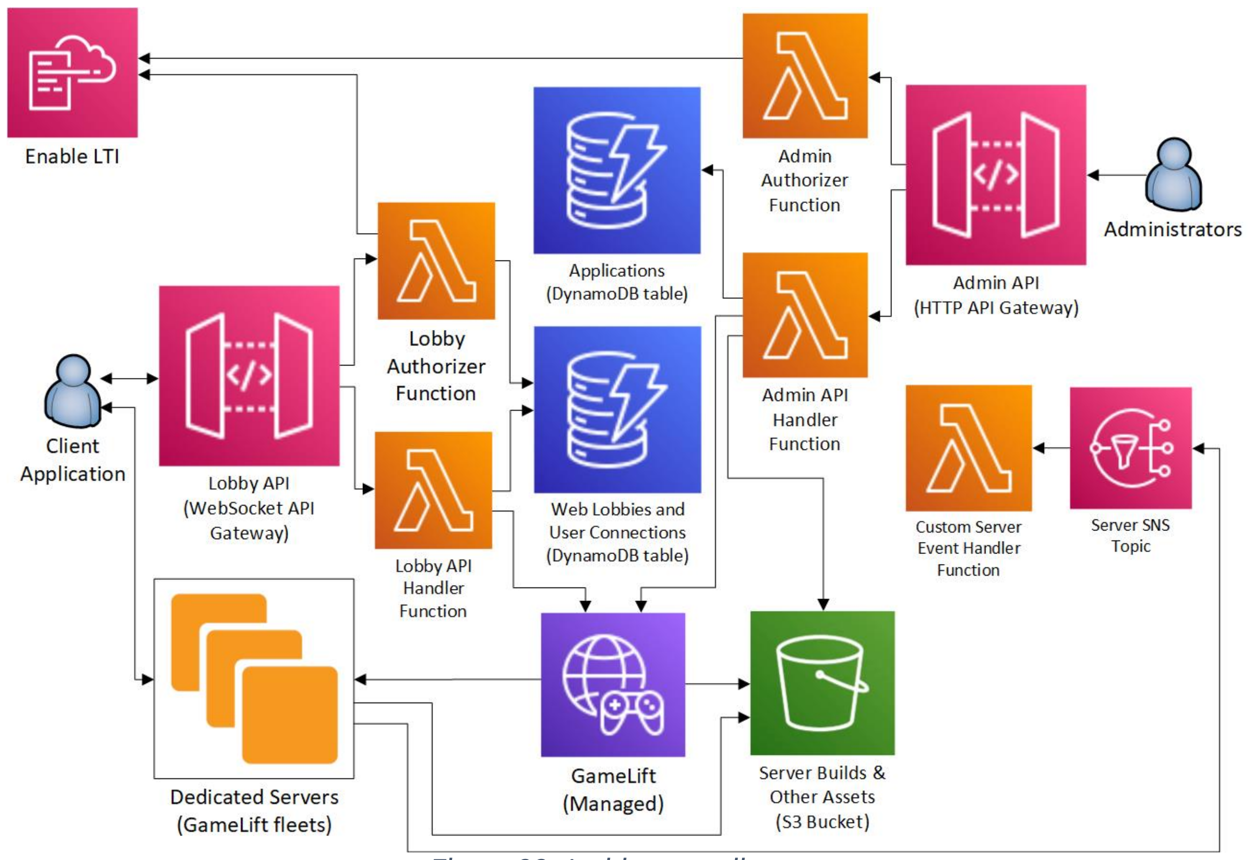
Lambda functions, which can then perform actions based on the event. In the AWS backend, the service is used for sending email notifications about critical events, and for decoupling the components of the architecture, making it more flexible and scalable.

- Lambda – A serverless compute service that allows developers to run code without the need to provision, manage, or scale servers. With Lambda, we can simply upload our code and AWS takes care of the underlying infrastructure. It supports a variety of programming languages, including Node.js, Python, Java, C# and Go. The service is designed to be event-driven, meaning it can automatically execute code in response to events such as incoming API requests or messages published to SNS. It provides several benefits, including automatic scaling, pay-per-use pricing, and high availability. The service is used for multiple purposes in the AWS backend, such as handling and responding to user requests, and sending email notifications about critical events.
- S3 – A scalable, highly available cloud storage service. In the AWS backend, it is used for storing the server binaries and other assets used by the applications which can be later downloaded to the fleet instances managed by GameLift.
- API Gateway – A managed service that allows us to create, publish and manage APIs such as REST or WebSocket APIs, and securely expose them to the internet in order to be called by client applications. APIs created with API Gateway can be easily integrated with other AWS services such as AWS Lambda. The service can also be used to create custom authorization and authentication mechanisms, as well as to throttle and monitor API usage. In the case of the AWS backend, web lobby and session-related services will be published through a WebSocket API, so that real-time notifications can be sent to the client applications about specific events. An HTTP API will provide functionalities for administrators to configure and install applications to the platform, and to specify fleet capacity limits per application.

8.1.1 Architecture Overview

The backend is based on a reference architecture suggested by Amazon, found in the GameLift documentation:
https://docs.aws.amazon.com/gamelift/latest/developerguide/gamelift_quickstart_customservers_designbackend_arch_serverless.html

The following architecture diagram () illustrates what resources the AWS backend consists of and how they communicate with each other. Each arrow points to the direction in which a request is initiated.



Enable LTI
Admin Authorizer Function
Administrators
Applications (DynamoDB table)
Admin API (HTTP API Gateway)
Lobby Authorizer Function
Client Application
Lobby API (WebSocket API Gateway)
Admin API Handler Function
Web Lobbies and User Connections (DynamoDB table)
Lobby API Handler Function
Custom Server Event Handler Function
Server SNS Topic
Dedicated Servers (GameLift fleets)
GameLift (Managed)
Server Builds & Other Assets (S3 Bucket)

Figure 23: Architecture diagram.

Each resource performs the following tasks:

- Lobby API – Exposes Web Lobby related functionalities to the clients through a WebSocket API such as creating and joining web lobbies.
- Lobby API Authorizer Function – When a client tries to establish a connection with the Lobby API, this Lambda Function is automatically invoked, checking whether the client is authorised to use the API by querying the LTI system.
- Lobby API Handler Function – Processes WebSocket requests sent to the Lobby API.
- GameLift – In managed mode, GameLift stores state information about active sessions and manages dedicated servers running on the GameLift fleet instances.
- DynamoDB – Stores a list of the active Web Lobbies, WebSocket connection ID of connected users, and data about deployed applications in a separate table.
- Server SNS Topic – Receives messages from dedicated servers and forwards them to the Custom Server Event Handler Function.
- Custom Server Event Handler Function – Processes messages sent by dedicated servers. This can be used for performing custom backend-related tasks, such as saving session data into a database at the end of a session. This may vary depending on the application.
- S3 Bucket – Stores dedicated server binaries and related application-specific assets so that they can be downloaded to the GameLift fleet instances or accessed by dedicated servers at runtime.
- Admin API – Provides functionality for administrators to install applications to the platform and to manually specify the number of maximum sessions for each application, through a HTTP API.
- Admin API Authorizer Function – Checks whether an HTTP request sent to the Admin API is originating from an authorised user with elevated privileges.
- Admin API Handler Function – Processes HTTP requests sent to the Admin API.

The lifecycle of a Web Lobby and a corresponding session consists of the following steps:

1. A user logs into the institution's Learning Management System (such as Moodle) from the browser, then, depending on where it is located in the LMS, clicks on a URL which redirects them to the e-DIPLOMA platform website. The URL redirection is performed by the Enable LTI component, which ensures that any user redirected to the e-DIPLOMA platform website will be properly authenticated.
2. After successful redirection, the user decides to enter the Web Lobby menu on the web page which causes the browser to try to establish a secure WebSocket connection with the Lobby API by sending an access token to it which was previously obtained from the Enable LTI system upon redirection.
3. The Lobby API Authorizer Function checks the validity of the access token using the Enable LTI component. If it is valid, then the WebSocket connection is successfully established, the Lobby API Authorizer Function saves the WebSocket connection ID into DynamoDB, and the user will be able to use the Lobby API from the browser. Otherwise, the connection request is rejected.
4. The user forms a Web Lobby with some other users using the Lobby API, and chooses one of the applications installed on the platform from a list. As a result, a request about creating a session for the given application will be sent to the Lobby API. The state and the member list of the Web Lobby are tracked in DynamoDB.
5. The Lobby API Handler Function checks whether there is enough capacity for creating a new session in the GameLift fleet belonging to the selected application, starts a session, then reserves a slot in that session for each user participating in the Web Lobby. For each slot, a Player Session ID is generated by GameLift, which is returned back to the corresponding users in addition to the connection information (such as IP address and port) through the WebSocket connection. The session ID generated for the Web Lobby is saved into DynamoDB as well.
6. When a user's browser receives a message containing the Player Session ID and the connection information, the browser launches the corresponding client application on the user's device, passing the message content to it as a command-line argument.
7. The client application connects to the dedicated server at the given IP address and port, then sends the Player Session ID to it.
8. The dedicated server checks the validity of the Player Session ID using GameLift. If it is valid, then the connection is successful, and any further messages sent between the dedicated server and the client are handled by the application (the messaging protocol depends on which game engine the server and client application have been built with). Otherwise, the user gets kicked from the dedicated server.
9. When the session is over, the dedicated server disconnects all clients and optionally sends a notification to the Server SNS Topic which can be used for triggering a custom Lambda function that can handle various backend-related tasks such as saving historical data about the session into a preconfigured database. After all these tasks have been performed, the dedicated server requests GameLift to terminate the session.
10. When a user is disconnected from the server, the client application is automatically closed, and the user is returned to the e-DIPLOMA platform website where they can decide whether they want to leave the Web Lobby, or stay in it and participate in further sessions.

8.1.2 Analytics

The platform will host an artificial intelligence recommender system designed to assist students throughout their course journey. To develop this system effectively, it is imperative to gather relevant data on students' interactions within the platform. Initially, the plan is to leverage student interactions

with LOs, as well as higher and lower-level elements. The recommender system will benefit from analysing all forms of student engagement, including interactions with files, videos, external links, and forums. Additionally, capturing data on the duration of these interactions, idle time, lesson completion time, and other related factors would be valuable.
All the aforementioned data collectively contribute to the recommendation process as students' progress through the course and generate new data. However, during the initial stages or when students are new and their capabilities and interaction patterns are unknown, it may be challenging to provide suitable recommendations. Therefore, it would be beneficial to augment the student's history with relevant data to address this issue. This additional data could encompass the student's mastery level in related subjects, learning preferences, and other relevant information. The goal is to develop a comprehensive understanding of the learner's characteristics to enhance our ability to predict their behaviours across different scenarios.
Among the various types of data that can be utilised, interaction data holds the utmost significance. Obtaining this data is relatively straightforward as it can be automatically logged whenever students interact with or access any learning resources on the platform.

8.2 DynamoDB schema

In this document, we do not reproduce a full reference for the database schema, but provide a visual overview of table relations. Please refer to the repository for the complete technical reference.

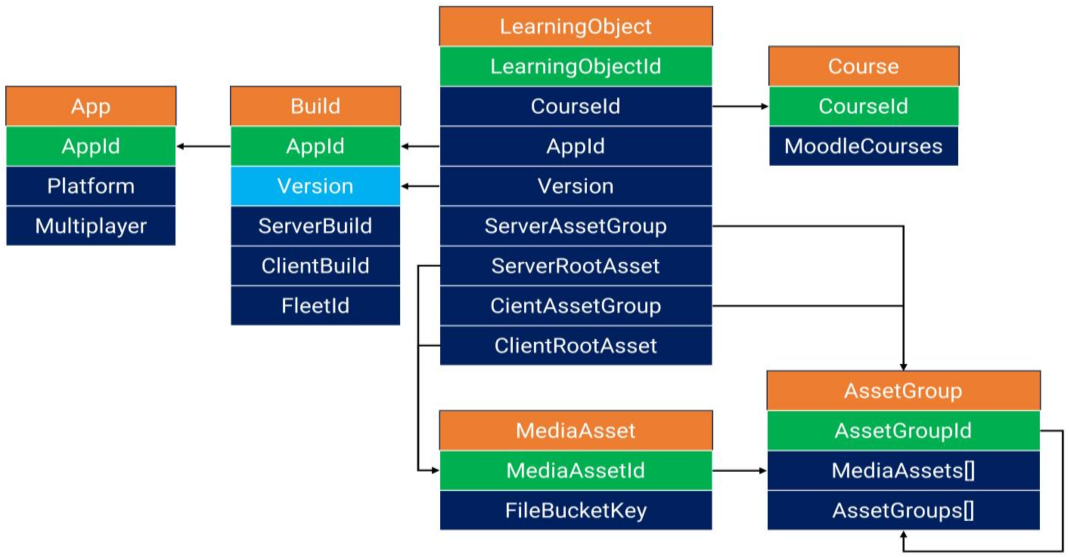


Figure 24: Simplified schematics of resource tables. Partition keys are in green, sort keys in light blue. Arrows point from one to many.

All resource tables have a Title, an Icon, an Owner, a Description, and a ConsentForms field (or equivalent). These are not depicted in Figure 24.

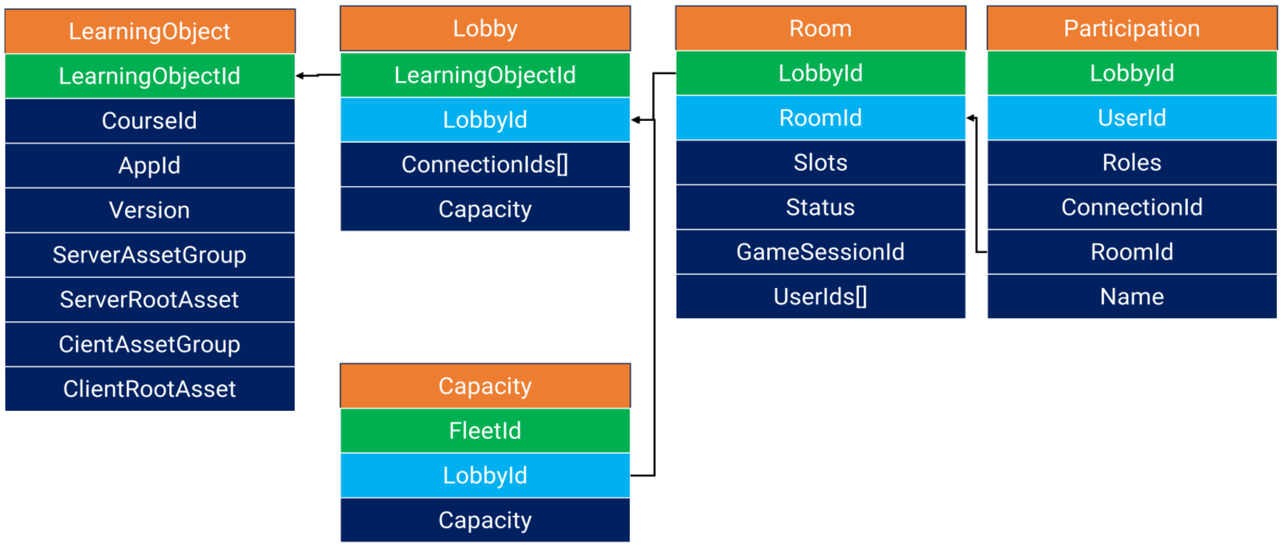


Figure 25: Tables for managing the Lobby.

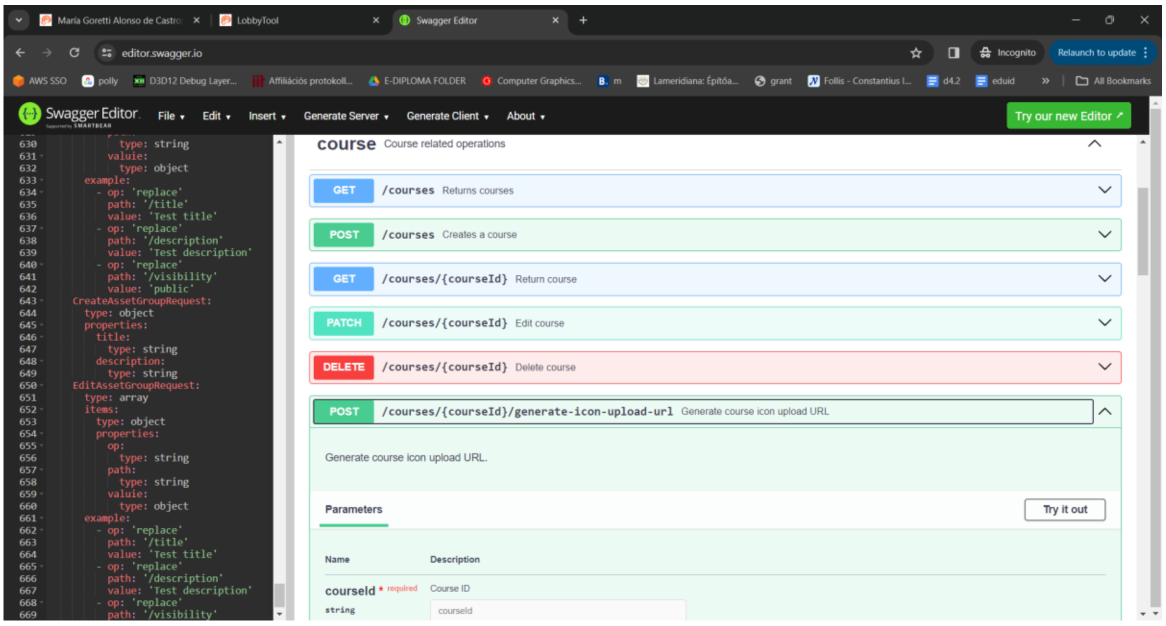
Figure 25 shows the main tables used for managing the Lobby. There may be multiple lobbies referencing the same learning object. We maintain both the total session capacity allocated against a fleet, and the per-lobby capacity. The Participation table tracks the users, while the Room table tracks room status.

8.3 Management HTTP API

The management API is a collection of HTTP endpoints facilitating listing, creating, and editing resource data. The API is documented in yaml files. For example, the GET /couses/{courseID} route is documented as:

```
/courses/{courseId}:
   get:
     tags:
       - course
     summary: Return course
     description: Returns a course.
     parameters:
       - name: courseId
         in: path
         description: Course ID
         required: true
         schema:
           type: string
     responses:
       '200':
         description: Successful operation
         content:
           application/json:
             schema:
               $ref: '#/components/schemas/GetCourseResponse'
       '404':
         description: Course does not exist
```

This data can be visualised using Swagger [27] in an interactive manner as shown in Figure 26. Please refer to the repositories for the Swagger documentation of the API.



```
        type: string
      valuie:
        type: object
  example:
    - op: 'replace'
      path: '/title'
      value: 'Test title'
    - op: 'replace'
      path: '/description'
      value: 'Test description'
    - op: 'replace'
      path: '/visibility'
      value: 'public'
CreateAssetGroupRequest:
  type: object
  properties:
    title:
      type: string
    description:
      type: string
EditAssetGroupRequest:
  type: array
  items:
    type: object
    properties:
      op:
        type: string
      path:
        type: string
      valuie:
        type: object
  example:
    - op: 'replace'
      path: '/title'
      value: 'Test title'
    - op: 'replace'
      path: '/description'
      value: 'Test description'
    - op: 'replace'
      path: '/visibility'
```

course Course related operations

GET /courses Returns courses

POST /courses Creates a course

GET /courses/{courseId} Return course

PATCH /courses/{courseId} Edit course

DELETE /courses/{courseId} Delete course

POST /courses/{courseId}/generate-icon-upload-url Generate course icon upload URL

Generate course icon upload URL.

Parameters Try it out

Name	Description
courseId * required string	Course ID courseId

Figure 26: Swagger visualization of the Courses HTTP API.

8.4 Lobby Websocket API

The Lobby API is a Websocket API, where messages can be send trough both directions. Here we list the API messages with their functions.

Table 5: Frontend => Backend messages:

Example message	Comment
`{ “type”: “getLobbyInfo” }`	Requests information about all rooms in the lobby. Triggered message: **lobbyInfo**.
`{ “type”: “createRooms”, “count”: 5, “slotsPerRoom”: 4 }`	Creates **[count]** number of rooms, each with a size of **[slotsPerRoom]** slots and **open** status. Triggered message: **roomsCreated**.
`{ “type”: “deleteRooms”, “roomIds”: [ “room-001”, “room-002”, “room-003” ] }`	Triggered message: **roomsDeleted**.
`{ “type”: “deleteAllRooms” }`	Triggered message: **roomsDeleted**.
`{ “type”: “lockRooms”, “roomIds”: [ “room-001”, “room-002”, “room-003” ] }`	Changes status of the selected rooms from **open** to **locked**. Triggered message: **roomStatusChanged**.
`{ “type”: “openRooms”, “roomIds”: [ “room-001”, “room-002”, “room-003” ] }`	Changes status of the selected rooms from **locked** to **open**. Triggered message: **roomStatusChanged**.
`{ “type”: “lockAllRooms” }`	Changes status of all **open** rooms in the lobby to **locked**. Triggered message: **roomStatusChanged**.

{
  “type”: “openAllRooms”
}
	Changes status of all **locked** rooms in the lobby to **open**.
	Triggered message: **roomStatusChanged**.

{
  “type”: “startRooms”,
  “roomIds”: [
    “room-001”,
    “room-002”,
    “room-003”
  ]
}
	Changes status of the selected **locked** rooms to **waiting**, and launches a session for each room.
	Triggered message: **roomStatusChanged**.

{
  “type”: “startAllRooms”
}
	Changes the status of all **locked** rooms in the lobby to **waiting**, and launches a session for each of those rooms.
	Triggered message: **roomStatusChanged**.

{
  “type”: “moveUserToRoom”,
  “userId”: “user-001”,
  “roomId”: “room-001”
}
	Moves a user to an **open** room. If the user is already participating in another **open** room, they will be automatically removed from it, otherwise, the request will fail.
	Triggered messages: **userJoinedRoom**, **userLeftRoom**.

{
  “type”:
“removeUserFromRoom”,
  “userId”: “user-001”
}
	Removes a user from an **open** room.
	Triggered message: **userLeftRoom.**

{
  “type”: “changeCapacity”,
  “capacity”: 1
}
	Allocates or relinquishes cloud capacity.

{
  “type”: “sendChatMessage”,
  “sender”: “user-001”,
  “addressees” : [
    “user-2”,
    “user-3”
  ],
  “addresseesRoleMask” : “♝”,
  “text” : “Hello!”
}
	A chat message was sent.

Table 6: Backend => Frontend messages:

Example message	Comment

{
  “type”: “lobbyInfo”,
  “capacity”: 2,
  “lobbyUsers”: [
	Status:
	● **open**: Joining or leaving the room is enabled,
	no session for the room has been launched.

Funded by
the European Union

{
  "userId": "user-001",
  "name": "Alice"
},
{
  "userId": "user-002",
  "name": "Bob"
},
{
  "userId": "user-003",
  "name": "Caleb"
}
],
"rooms": [
  {
    "roomId": "room-001",
    "slots": 4,
    "status": "locked",
    "participants": [
      "user-001",
      "user-002",
      "user-003",
      "user-004"
    ]
  },
  {
    "roomId": "room-002",
    "slots": 4,
    "status": "open"
    "participants": [
      "user-005",
      "user-006",
      "user-007"
    ]
  }
]
}
	• **locked**: Joining or leaving the room is disabled,
	no session for the room has been launched.
	• **waiting**: Joining or leaving the room is disabled,
	a session for the room has been launched, but has not yet started, participants are waiting.
	• **starting**:
	• **active**: Joining or leaving a room is disabled,
	a session for the room has been started, session information is available.

{
  "type": "roomsCreated",
  "slotsPerRoom": 4
  "roomIds": [
    "room-001",
    "room-002",
    "room-003"
  ]
}
	New rooms have been added.

{
  "type": "roomsDeleted",
  "roomIds": [
    "room-001",
    "room-002",
    "room-003"
  ]
	Rooms have been removed.

Funded by the European Union

}	

{
  “type”: “roomStatusChanged”,
  “rooms”: [
    {
      “roomId”: “room-001”,
      “status”: “locked”,
    },
    {
      “roomId”: “room-002”,
      “status”: “locked”,
    }
  ]
}
	Some rooms changed status.

{
  “type”: “userJoinedRoom”,
  “userId”: “user-001”,
  “roomId”: “room-001”,
  “movedBySupervisor”: true
}
	A user joined a room.

{
  “type”: “userLeftRoom”,
  “userId”: “user-001”,
  “roomId”: “room-001”,
  “movedBySupervisor”: false
}
	A user left a room.

{
  “type”: “userConnected”,
  “userId”: “user-001”
}
	An earlier user has reconnected.

{
  “type”: “userDisconnected”,
  “userId”: “user-001”
}
	A user can no longer be verified to be connected to the lobby.

{
  “type”: “capacityChanged”,
  “capacity”: 2
}
	The cloud session capacity has changed.

{
  “type”: “newLobbyUser”,
  “userId”: “user-001”,
  “name”: “Alice”
}
	A user entered the lobby.

{
  “type”: “sessionInfo”,
  “address”: “198.51.100.12:7777”,
  “playerSessionId”: “psess-001”
	A game session has started.

“launchInfoUrl”: “https://………” }	

{
  “type”: “chatReceived”,
  “sender”: “user-001”,
  “addressees” : [
    “user-2”,
    “user-3”
  ],
  “addresseesRoleMask” : “♝”,
  “text” : “Hello!”
}
	A chat message has arrived.

8.5 The e-DIPLOMA Frontend

The Frontend is an Angular Typescript web application. Node.js and npm have to be installed on the development computer. After `npm install`, the web pages can be built with `ng build`. The result of the build must be server on the web for the webpages to be accessible. This is done by uploading the distribution to an S3 bucket. Building and deploying the pages has been automated in the `deploy.bat` script. The repository also contains a Sublime Text project with a build system that build and deploys the entire frontend with one Ctrl+B key combination. Note that for this to work, command line access to AWS has to be established through `aws sso configure`, and the appropriate role settings in AWS IAM Identity Center.

8.6 The e-DIPLOMA Launcher

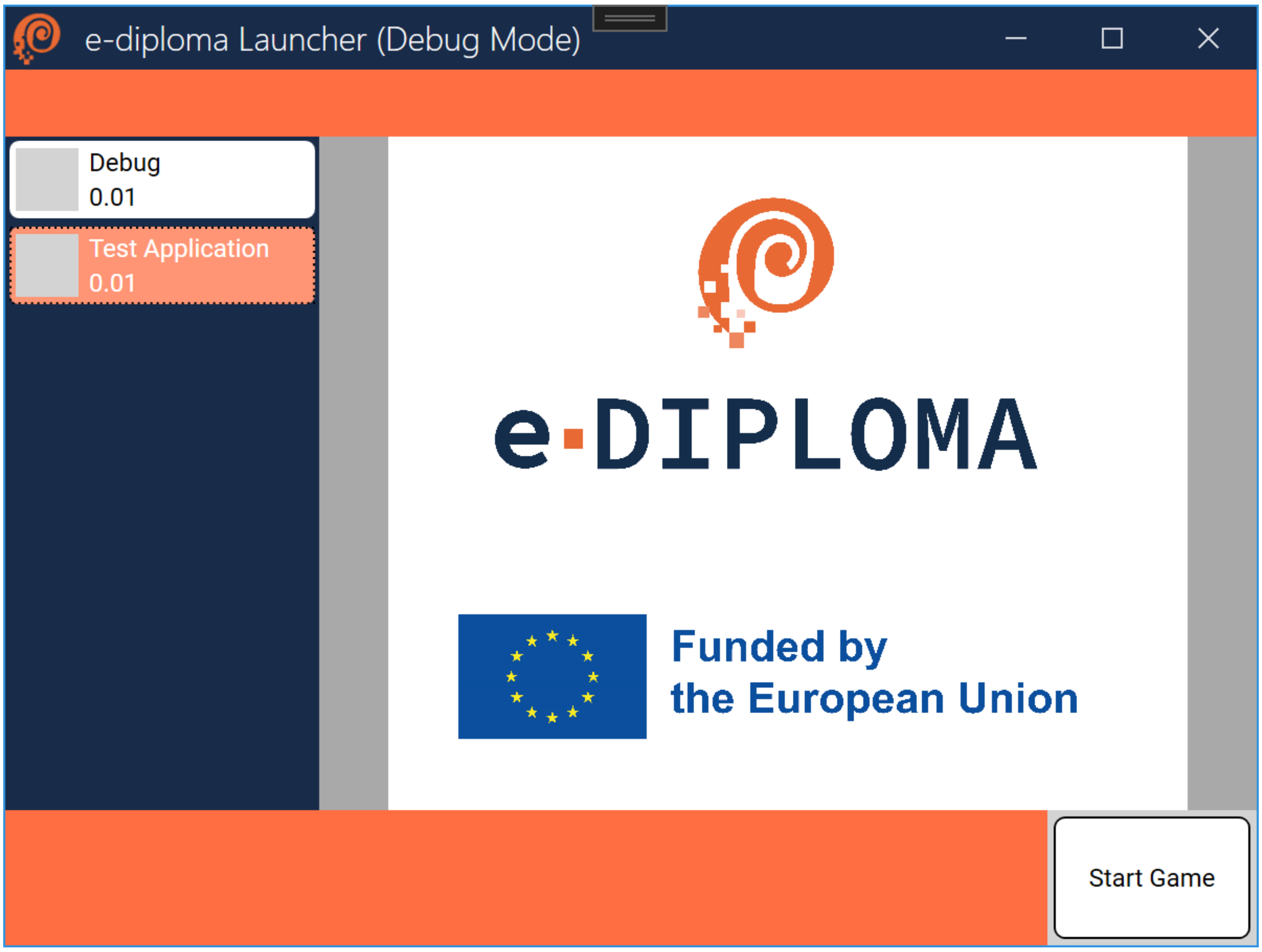


Figure 27: e-DIPLOMA launcher's user interface

The Launcher is Windows **.NET 6.0** desktop application written in C#. It makes use of both **Windows Forms** (for single instance functionality) and **WPF** for the user interface. The Microsoft Visual Studio 2022 project is available in the repository at https://github.com/BME-IIT-CG/e-DIPLOMA-Launcher. The following libraries were used:

- CommandLineParser
- MahApps.Metro and MahApps.Metro.IconPacks
- Newtonsoft.Json

The project requires **Microsoft Studio Installer Projects 2022** to be opened properly.
The program is designed to allow users to install, update, configure, uninstall and launch various other applications integrated into the e-DIPLOMA platform. The application may be opened via the e-DIPLOMA Lobby website which opens a custom custom URI (ediploma:). This URI scheme is registered by the Launcher Installer in the Windows registry as

```
HKEY_CLASSES_ROOT/ediploma

    (Default) = "URL:ediploma"

    URL Protocol = ""

    shell/open/command

            (Default) = "{path to application}" "%1".
```

The Lobby also passes the parameters required to connect to the backend and download appropriate educational games and assigned assets. The parameters are:

4. *OperationRaw*: Specifies the action to perform with the selected game in the launcher. It accepts values from the *GameOperation* enum. If provided, the version must also be specified. Values may be:
   NONE: Just open the launcher, and select the game, if `AppStoreEndpointUrl` and `AccessToken` are provided.

   DOWNLOAD: Open the launcher and download the game and it's assets if needed,

   LAUNCH: Open the launcher and launch the game, if possible. If not, attempt a download and allow a launch later.
5. AppStoreEndpointUrl: Specifies the URL of the game descriptor and download location. Needed for `DOWNLOAD`.
6. AccessToken: Specifies the access token required for downloading the game and assets. This token is valid for 30 minutes. Needed for `DOWNLOAD`.
7. SessionAddress: Specifies the session address to connect to using the session ID. If provided, both it and `PlayerSessionId` must also be specified. Needed for `LAUNCH`.
8. PlayerSessionId: Specifies the player ID to connect to the session. If provided, both it and `SessionAddress` must also be specified. Needed for `LAUNCH`.

The launcher downloads games and assets into the user's temp directory. Once the downloads are complete, games and assets are extracted into their own subdirectories, inside the launcher's installation directory (by default `assets` and `games`). Elevated permissions may be required when the default installation path for the launcher has been used (Program Files). The list of games and assets are kept in two JSON files: `games.json` and `assets.json`. The launcher can launch any runnable file and also pass a number of static arguments provided, all taken from the JSON. The launcher also sets the working directory to the folder of the executable and set the following local environment variables:

- "PLAYER_SESSION_ID",
- "SESSION_ADDRESS",
- "ROOT_MEDIA_ASSET_ID", if there there is at least one asset provided to the game.

Games may also have a background image and an icon shown in the launcher. They must be named `background.png` and `icon.png` respectively, and be placed inside the root folder of the game.

8.7 Virtual production tool Launcher

In order to allow the e-Diploma platform to deploy, on demand, the Virtual production tool to carry out immersive and interactive learning sessions, the EC2 technology of the AWS ecosystem has been employed. This methodology has been chosen because of the following. As introduced in section 8.1, the virtual production tool application needs to be virtualized due to its requirements in terms of hardware requirements, especially graphical acceleration. The allocation of this tool on cloud premises will avoid the need for professors and students to have a local machine with powerful processing capabilities and allow them to connect and manage the tool even employing mobile smart devices. The virtual production tool launcher was supposed to be deployed employing the GameLift, the main methodology in e-Diploma Platform to leverage virtualized tools. The issue found trying to implement the launcher with GameLift has been related to the images available for this AWS method. BRA reviewed all the images available, in terms of CPU and GPU capabilities, for GameLift. The result of this prospection revealed that in the AWS ecosystem, the base images that can support the requirements of the virtual production tool are only available in EC2. Minimum requirements for Edison software are defined in Table 7.

Table 7: EMI Edison requirements.

Instance Size	GPU	vCPUs	Memory (GiB)
g4dn.2xlarge	1	8	32

For this reason, also the EC2 system has been included in the platform, as reported in the architecture overview section Architecture Overview. This step required an added effort for the platform implementation from a time, effort and costs point of view. Selected the methodology to leverage the deployment and the base image (AMI) to prepare the tool deployment (windows based g4dn.2xlarge), the SW assets to be installed are mainly of two categories: Edison and AWS related. For the e-Diploma virtual production tool image the following applications and tools have been installed:

- Edison 5.1
  - pdf exporter plugin
  - PPT exporter plugin
  - Video grabber tool (OBS)
  - MS Teams
  - Unreal Engine (Optional. It requires superior base image capabilities)
- AWS
  - NiceDCV client

The last-mentioned tool related to the AWS ecosystem will be the bridge that will permit the connection to the virtual machine from different devices and tools. This tool, in case the professor will use the green screen feature, will bring the video and audio signal from the professor's local machine to the virtual machine where this signal is processed by Edison to embed the professor in the virtual environment. Apart from the professor's video signal, the other assets that need to be transferred to the virtual machine are all the assets needed to prepare the content. As introduced in section 7.1, the first step for the professor is to employ the virtual production tool to prepare the educational session. As depicted in the

figure at the beginning of Architecture overview section, the content is stored in the platform thanks to S3 Buckets.
As per the steps described in section 7, the professor will select the content to transfer to the Edison VM. This content will then be available to be employed by the Edison Quick start wizard (Figure 28).

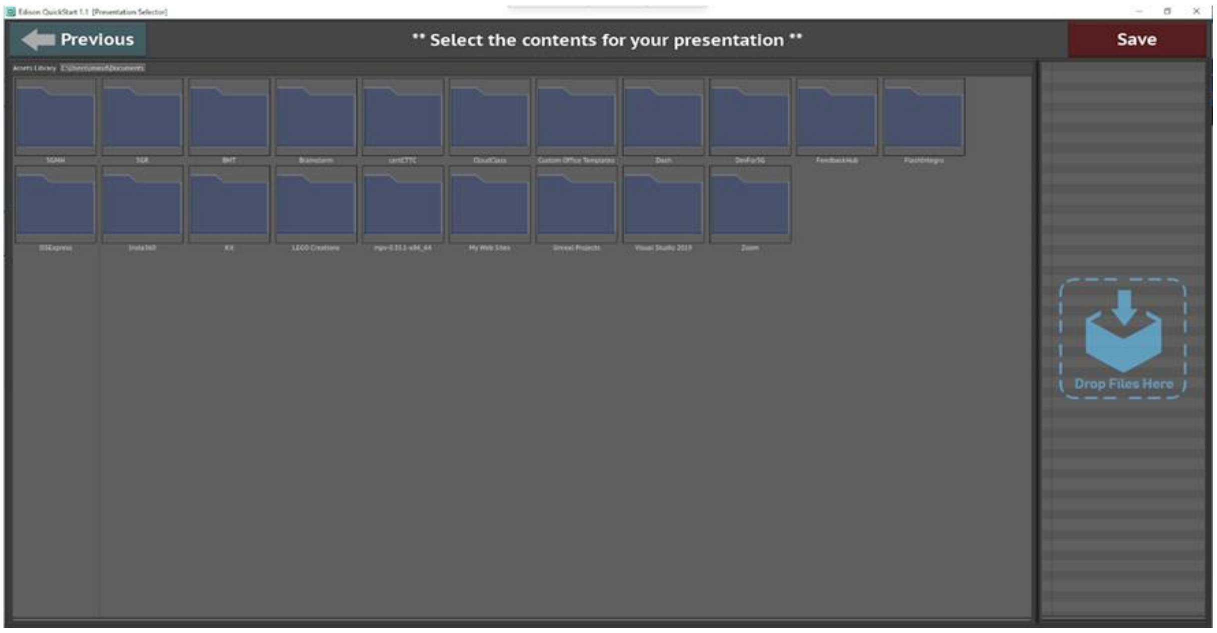
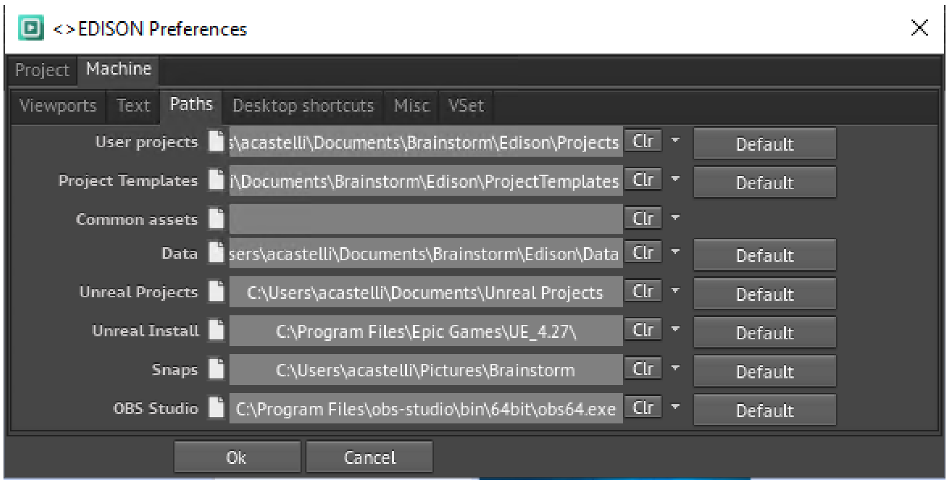


Figure 28: Edison QuickStart interface.

In order to accomplish this, the contents transferred should be copied in the following folders (Figure 29).



User projects	s\acastelli\Documents\Brainstorm\Edison\Projects	Clr	Default
Project Templates	i\Documents\Brainstorm\Edison\ProjectTemplates	Clr	Default
Common assets		Clr	
Data	sers\acastelli\Documents\Brainstorm\Edison\Data	Clr	Default
Unreal Projects	C:\Users\acastelli\Documents\Unreal Projects	Clr	Default
Unreal Install	C:\Program Files\Epic Games\UE_4.27\	Clr	Default
Snaps	C:\Users\acastelli\Pictures\Brainstorm	Clr	Default
OBS Studio	C:\Program Files\obs-studio\bin\64bit\obs64.exe	Clr	Default

Figure 29: Edison folders configuration tool.

Once the Edison project is created and saved in the VM, when the user decides to close the session of the virtual production tool, the edited project stored at the previously introduced folder location is transferred back to the S3 bucket space of the user that accessed the instance. The instance is then closed without the need to save anything simplifying the maintenance of the platform from a persistency point of view.

When the professor will need the edited Edison project to carry out the class or the content generation, the following session virtual production tool session, once it asks for the virtual production tool deployment, the saved projects, stored at S3 Bucket user space, are copied back to the Edison VM.

9 Source code and deployed service access

The source code is available in the repositories on GitHub, under the organisation BME-IIT-CG (https://github.com/BME-IIT-CG), using the personal fine-grained access token below:

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

This token gives read access to the repository contents and write access to Issues.

The deployed service is available through Moodle, at https://cg.iit.bme.hu:3004. The following users have been registered to facilitate evaluation:
Student accounts:
- zutano/Review+2
- novak/Review+2

Teacher accounts:
- hamares/Review+2
- malonsca/Review+2

10 Conclusions

This document accompanies the software constituting deliverable D4.2 of the e-DIPLOMA project, which is delivered through repositories and access to the cloud-deployed platform. We described the work done on the e-DIPLOMA platforms, specified the technical details, and gave a detailed overview of functionality and usage. The platform serves as the basis of further development in WP4, creating applications that on one hand support the research goals of the prototype courses, and on the other hand realise the potential of the e-DIPLOMA platform by employing the features demonstrated in this document in production.

11 References

[1] Amazon DynamoDB. [Online]. Available: https://aws.amazon.com/dynamodb/#:~:text=Amazon%20DynamoDB%20is%20a%20fully,data%20import%20and%20export%20tools.

[2] Secure Cloud Services. [Online]. Available: https://aws.amazon.com/pm/ec2/

[3] Enable LTI. [Online]. Available: https://aws-samples.github.io/enable-lti/

[4] Adding a Tool Site-wide. [Online]. Available: https://docs.moodle.org/402/en/External_tool_settings#Adding_a_tool_site-wide

[5] Amazon S3. [Online]. Available: https://aws.amazon.com/s3/

[6] Unreal Engine. [Online]. Available: https://www.unrealengine.com/en-US/unreal-engine-5

[7] Unity3D. [Online]. Available: https://unity.com/

[8] eduID. [Online] Available: https://eduid.hu/en/

[9] Amazon GameLift. [Online.] Available: https://aws.amazon.com/gamelift/

[10] Fusion Documentation: https://doc.photonengine.com/fusion/current/getting-started/fusion-intro

[11] Fusion Dedicated Server: https://doc.photonengine.com/fusion/current/manual/dedicated-server-overv iew

[12] AWS GameLift: https://aws.amazon.com/gamelift/

[13] GameLift SDK Documentation for Unity and C#: https://docs.aws.amazon.com/gamelift/latest/developerguide/integration-server-sdk5-csharp.html

[14] GameLift Unity Integration: https://docs.aws.amazon.com/gamelift/latest/developerguide/integration-engines-unity-using.html

[15] GameLift Local CLI: https://aws.amazon.com/gamelift/getting-started/

[16] The tutorial I used to learn Fusion: https://www.youtube.com/watch?v=KqpMOdPj3co&list=PLyDa4NP_nvPfHhPuumJylSj8jXyULsT1X&pp=iAQB

[17] Fusion libs: https://doc.photonengine.com/fusion/current/getting-started/sdk-download 9. GameLift Server SDK binaries: https://gamelift-server-sdk-release.s3.us-west-2.amazonaws.com/csharp/Ga meLift-CSharp-ServerSDK-4.0.2.zip

[18] Documentation MoodleDocs. Available at: https://docs.moodle.org/403/en/Main_page (Accessed: 29 February 2024).

[19] Sakimura, N., Bradley, J., Jones, M., De Medeiros, B., & Mortimore, C. (2014). Openid connect core 1.0. The OpenID Foundation, S3.

[20] Hughes, J., & Maler, E. (2005). Security assertion markup language (saml) v2. 0 technical overview. OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08, 13, 12.

[21] Learning tools interoperability(lti)® advantage implementation guide 1.3 1edtech final release (no date) Learning Tools Interoperability(LTI)® Advantage Implementation Guide 1.3 1EdTech Final Release | IMS Global Learning Consortium. Available at: https://www.imsglobal.org/spec/lti/v1p3/impl/ (Accessed: 29 February 2024).

[22] HTML5. Available at: https://www.w3.org/TR/2011/WD-html5-20110405/index.html (Accessed: 29 February 2024).

[23] Angular. Available at: https://angular.io/docs (Accessed: 29 February 2024).

[24] TypeScript. Available at: https://www.typescriptlang.org/docs/handbook/ (Accessed: 29 February 2024).

[25] Ni SP support (2023) NI SP. Available at: https://www.ni-sp.com/support/ (Accessed: 29 February 2024).

[26] https://datatracker.ietf.org/doc/html/rfc3339

[27] API development for everyone. Swagger. Available at: https://swagger.io/ (Accessed: 29 February 2024).

e-DIPLOMA

Funded by
the European Union