ADA University
School of Information Technology and Engineering

Senior Design Project

Final REPORT

Project title: ***"Integrating Unity 3D Game Engine and Anthology Blackboard Learning Management System."***

Project Advisor: **Dr. Araz Yusubov**

Authors:

1. [IT] Sabina Veyisli
2. [IT] Fatima Khalifali
3. [IT] Lala Mahmudova
4. [IT] Tangiz Alizada

Industry Mentor: [ADA] Tural Hamzayev

Baku, May 2023

List of Abbreviations

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
ARCore	Augmented Reality Core
ARKit	Augmented Reality Kit
BB	Blackboard
D2L	Desire2Learn
HTC	High Tech Computer
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
iOS	iPhone Operating System
JSON	JavaScript Object Notation
LMS	Learning Management System
LSL	Linden Scripting Language
MOODLE	Modular Object-Oriented Dynamic Learning Environment
REST	Representational State Transfer
RM	Reference Model
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SL	Second Life
SLOODLE	Second Life Object-Oriented Distributed Learning Environment
SRP	Single Responsibility Principle
STEM	Science, Technology, Engineering and Mathematics
UWP	Universal Windows Platform
VLE	Virtual Learning Environment
VR	Virtual Reality
WebGL	Web Graphics Library
XML	Extensible Markup Language
2D	2 Dimensional
3D	3 Dimensional

Abstract

The purpose of this project is to develop an importable Unity package for game developers to integrate Blackboard with Unity. The integration of Blackboard with Unity will offer an efficient way for faculty and students to collaborate, communicate, and divide their works within the context of the game. The project makes practical and effective use of UnityWebRequest to create communication between Unity and Blackboard, and the implementation is done through using C# programming language and REST API. This project offers potential benefits both for educational institutions and students by enhancing e-learning in an interactive and engaging way. As a result, this project can remarkably impact the game-based learning field and can be implemented into different educational environments.

Keywords: Unity Game Engine, Blackboard Learning Management System, game-based learning, importable unity package, C#, REST API, UnityWebRequest.

INTRODUCTION

1.1 Definition

In the world of gamification, incorporating educational content and interactive elements often presents a complex challenge. The package the team developed is an effective tool that functions as an intermediary between Unity 3D and Blackboard Learn, allowing for smooth integration of the two systems. With this package, developers can concentrate only on building immersive gaming worlds within Unity, while the package manages the connection of endpoints and game objects automatically.

The aim of the project is to simplify the process of connecting game objects in Unity 3D to Blackboard Learn endpoints, making it easier to integrate educational content and interactive elements into simulated environments. The package provides developers with a user-friendly solution that saves time and effort by abstracting the complex nature of working with the Blackboard Learn API.

What distinguishes this Unity package from others is its emphasis on usability and automation. While other solutions may necessitate manual endpoint assignment and connection, our package automates this process, eliminating the need for developers to write extensive code or configure complex settings. This automation speeds up the development process and reduces errors, allowing developers to focus on creating engaging gameplay experiences.

1.2 Project Purpose

The purpose of the project is to bridge the gap between traditional learning management systems (LMS) and modern gaming technologies. As the educational landscape continues to evolve, there is an increasing demand for innovative ways to engage students and promote active learning. By incorporating gaming elements into the learning process, developers can create a more immersive and enjoyable experience that enhances student motivation and knowledge retention.

Furthermore, the primary reason behind this goal is to divert learners' attention away from traditional learning methods and contribute to the development of new interactive learning approaches. Traditional instruction and solitary knowledge acquisition have clearly failed to capture the interest of young people and cannot fully address the academic needs of society. Each student's learning pace and style are different, and it is difficult for professors to deliver individual directions to each student using the traditional face-to-face teaching approach.

Overall, the purpose of this project is to facilitate the merge of gaming and education, and thereby offer a novel and exciting way for students to learn and grow in their academic pursuits.

1.3 Project objectives

The project's primary objective is to provide developers with an importable package that makes it easy to build a fun and interactive environment by integrating Blackboard Learn and Unity 3D Game Engine.

By providing developers with a seamless connection between Blackboard Learn and Unity 3D, this package allows for the creation of customized games and simulations. This will not only benefit students by making the learning experience more engaging but also give educators an opportunity to tailor the curricula to the unique needs and interests of each student.

Moreover, this project will also pave the way for the integration of emerging technologies, such as virtual reality and augmented reality, into the learning process. As these technologies become more mainstream, it is important to provide a framework for their integration and use within an educational context.

Our package is novel in that it is the first to connect Blackboard Learn and Unity 3D in this manner. The package's key features include the ability to read data from Blackboard, such as student usernames, and course content, and use that data to create personalized game experiences. It allows developers to create games that take advantage of Blackboard's learning content, announcements, grades, and other learning materials. Furthermore, it offers the flexibility to create custom games from scratch, enabling developers to integrate unique learning experiences for their users.

Blackboard Learn is a well-known learning management system that is used for online teaching, learning, community building, and knowledge sharing [8]. It has been in use in academic institutions for approximately 23 years, and it supports various functions such as course administration, content creation, and grading workflows [8]. In terms of online teaching, Blackboard Learn's Blackboard Collaborate functionality played a significant role, particularly during the pandemic years.

Unity is a game engine that supports 2D and 3D visuals and C# programming. Essentially, Unity has physics built-in so that developers don't have to worry about every detail. For instance, Unity offers an Asset Store where developers can publish their own works and download those of other Unity developers. In the next sections of this paper, more detailed information about the components of the project will be given.

1.4 Project Significance and Novelty

Gamification is defined as “the use of game mechanics and experience design to digitally engage and motivate people to achieve their goals” [3]. In today’s world, there are numerous gamified applications in use in sectors such as healthcare, finance, and, most notably, education. Gamification plays a vital role in increasing engagement in the education field. It is an established fact that a game-like atmosphere helps to increase student productivity [16]. Creating and successfully completing challenges increases students' academic motivation and, as a result, directly improves their academic excellence too [19].

Many scholars have investigated the impact of gamification in an educational context, yielding positive results such as increased engagement, user retention, knowledge, and cooperation [3]. In an experiment led by University of Bells instructors, two experimental groups (gamified and non-gamified) were formed in which learners used gamified and non-gamified versions of Feeper (a popular LMS in Brazil) respectively [16]. Feeper's gamified version featured only points, badges, and ranks as game mechanics. According to this research, participants who used a gamified environment had greater average points, badges, and ranks at the end of the semester than learners who used a non-gamified environment. It is demonstrated that gamified group participants had a significant improvement in terms of the quality and accuracy of submitted solutions [16].

There are numerous psychological factors that explain how gamification influences brain processes, particularly the reward center [13]. When a student wins or receives immediate positive feedback, it modifies the brain's reward function. Individual motivation is directly tied to this reward function. Human motivation is classified into two groups: Intrinsic and extrinsic motivation [15]. Intrinsic motivation includes doing something because it personally benefits you and your desire. Extrinsic motivation, on the other hand, includes performing something to earn a reward. In other words, when it comes to gamification in education, intrinsic motivation occurs when a student is satisfied and appreciates the subject itself, whereas extrinsic motivation occurs when a learner expects to obtain a tangible reward for completing a task, such as badges or high scores on a scoreboard [15]. It is stated that removing common extrinsic rewards such as points, badges, and leaderboards reduces students’ motivation.

1.4 Problem Statement

Many academic programs have high attrition rates and low degree completion rates, which are often linked to the lack of intrinsic motivation among students [16]. Implementing gamification tools can help increase student productivity and engagement in education. However, there may be challenges in gaining authorization to innovate and implement modern teaching methods. Gamification and game-based learning are different aspects, with the latter making learning more contextual and enjoyable for students. The goal is not only to address traditional teaching issues but also to promote the use of game-based learning and gamification in education.

According to a 2013 survey by the US Department of Education, in STEM (Science, Technology, Engineering, and Mathematics), 48% of bachelor's degree candidates and 69% of associate degree candidates quit the field of study or dropped out altogether [8]. It is argued that this is directly linked to the lack of intrinsic motivation, and by implementing the right tools of gamification, students will be more productive and willing to education.

Most schools continue to suffer in using modern teaching methods; they often experience low engagement problems. It is worth noting that higher college courses frequently experience fewer engagement issues because they are interested in implementing gamification tactics and developing long-game mechanisms [8]. But game mechanics themselves, may also not be enough for solving low engagement problems. Although professors want to use gamification and other modern instructional methods, there are some problems regarding the authorizations from educational departments to innovate.

It is worth noting that gamification and game-based learning are completely different aspects. Gamification, as mentioned before, is the mixture of game elements into a learning environment [2]. Game-based learning makes learning contextual, more real, and therefore more enjoyable and relevant to learners. A virtually improved world invites students to use their imaginations and explore new possibilities in an interactive setting. The purpose, in that sense, is not only to cope with the main issues of traditional teaching but also to help popularize the use of game-based learning and gamification in the field of education [14].

2. LITERATURE REVIEW

The integration of educational platforms with the development of games has attracted substantial interest in recent years because of its potential to improve learner engagement, knowledge acquisition, and cooperation. As the need for immersive and interactive educational experiences grows, developers are looking for new ways to seamlessly integrate game environments with instructional content. In this section, we conduct a literature review to look into existing options for integrating popular LMSs with Game environments.

The purpose of this literature research is to obtain an understanding of the current state of integrating Blackboard Learn with Unity 3D and to identify the strengths and limitations of existing solutions. We intend to emphasize the distinct features and benefits of various approaches by reviewing prior research and industry practices. This review will assist us in positioning our Unity package, which serves as a bridge between Blackboard Learn and Unity 3D, within the existing body of work, as well as demonstrating how our solution meets the highlighted gaps and issues.

2.1 Sloodle

Sloodle is a free and open-source project that intends to connect multi-user virtual environments, notably Second Life, with the Moodle LMS [17]. Sloodle is a virtual classroom environment that promotes rich and interactive learning experiences by merging the virtual world of Second Life with the capability of Moodle [17].

Sloodle's ability to link Virtual Learning Environment (VLE) technologies with the immersive and interactive nature of Second Life is one of its primary characteristics. Sloodle allows users to communicate with their classmates in the form of avatars in a virtual classroom. The platform allows a variety of communication methods, including text and voice chats, which are enabled by the Moodle chat room [17]. Furthermore, Sloodle allows users to view slides and documents within the virtual environment via virtual boards, allowing for real-time presentations. These resources can also be posted asynchronously to Moodle in a variety of formats for later access and review.

Sloodle has different features that let students improve their learning experience in virtual classrooms. It, for example, includes a game scoreboard that monitors points, allowing gamification components to be included in educational sessions [17]. Sloodle also includes a quiz chair feature, which allows students to take quizzes directly within the virtual environment. The technology also allows educators to manage scenarios with several objects, increasing the overall versatility and customization possibilities accessible to them.

Sloodle was created with the PHP scripting language and consists of a collection of PHP scripts maintained on a web server with a database backend. Sloodle can be accessed via a web browser, which communicates with the server via the HTTPS protocol. Similarly, when users use the Second Life client to interact with data in the Moodle database, the communication is accomplished via HTTP requests made from objects within Second Life and handled by Sloodle modules on the server.

Sloodle, however, has had obstacles and limits since its initial appearance, despite its original promise. In 2011, the project was updated, however, after that time, there were no significant improvements on this project [17]. One of the key difficulties, according to sources, was the strict limits on the quantity of data that could be supplied in a single request and returned in a response. Furthermore, at the time, the scripting language used in Second Life (LSL) did not fully support HTTPS and lacked cookie support, making it difficult to maintain user sessions within the integration.

2.2 Unity + Blackboard Learn

Another credit should be given to a project named Unity + Blackboard Learn which establishes a direct connection between Unity and BB Learn via REST API [23]. In this specific project, it is possible to give a functionality to a certain GameObject in Unity that represents a certain endpoint of BB Learn with the help of GET/POST **methods of REST API,** thus proving the possibility of integrating Blackboard LMS into a game realm.

2.3 Unity + SCORM

Understanding “what is SCORM?” is essential for trainers and developers who need to decide what role it should play in their learning programs. The Sharable Content Object Reference Model is abbreviated as SCORM. The term RM (Reference Model) describes that SCORM is a standard; it is a technical specification or set of instructions that can be used to standardize the way eLearning courses are created called SCORM-compliant courses [21]. SCO (Sharable Content Object) describes the elements of the SCORM package that can be reused across multiple platforms [21]. In simple words, there is a SCORM package as a zip file that can be understood by compatible LMS that are also SCORMcompliant.

One example of a similar system is Unity and SCORM Integration Kit for WebGL (Web Graphics Library). This system intends to create a Unity package and load it into various LMSs such as BB LMS, Moodle (, D2L (Desire2Learn), or Canvas as course content, assignment, etc. When the student opens the content created in Unity + SCORM package the LMS connects to and interacts with the scene/game created in Unity via ScormAPIWrapper. The student’s score, time spent on completion, the status of completion, and whether he passed/failed a certain assignment/task/exam will be visible in the

LMS once the student exits the scene/game. To sum up, SCORM is not an LMS, it is standard, and standards are important in any field. Today, almost any LMS recognizes SCORM.

As a conclusion, there are only a few projects intended to build an importable package for developers that supports the integration of gamification and works as a bridge between LMS and Game Development Tools. The main problem is that many projects are focused on implementing plug-ins to the already existing system or integrating SCORM, rather than building a package that supports the integration of a game with LMS using REST API. While the most similar system Unity and Blackboard LMS project by Michael Bechtel succeeded in establishing a preliminary connection between Unity and Blackboard LMS, it also lacks in terms of integrating certain endpoints of Blackboard LMS to the game environment as well as convenience and usability, which distinguishes it from the project. Building a more convenient importable package for developers to further connect Blackboard LMS to Unity games and step into a gamified learning environment.

2.4 The Differences

Our project, which focuses on developing a Unity package that connects Blackboard Learn endpoints to game objects in Unity 3D, differs from existing solutions such as Sloodle, Unity + Blackboard Learn, and Unity + SCORM Integration Kit for WebGL.

To begin, unlike Sloodle, which merges Second Life and Moodle, our project focuses on the connection between Blackboard Learn and Unity 3D. While Sloodle provides a virtual classroom and communication tools within Second Life, our package aims to bridge the gap between game development and Blackboard Learn, allowing developers to create immersive game environments while seamlessly incorporating educational content.

In comparison to the Unity + Blackboard Learn project, our solution is completer and more automated. While the Unity + Blackboard Learn project connects Unity and Blackboard Learn via REST API, it primarily focuses on the functionality of specific game objects that represent Blackboard Learn endpoints. Our package goes above and beyond by automating endpoints and user authorization, acting as a bridge between Unity and Blackboard Learn. This simplifies the integration process for developers, allowing them to concentrate on developing the gaming environment while our package handles the endpoint connections.

Furthermore, unlike the Unity + SCORM Integration Kit for WebGL, our project takes a different approach to integration. While the Unity + SCORM Integration Kit focuses on loading Unity packages into various SCORM-compliant Learning Management Systems (LMS), our package focuses on Blackboard Learn specifically. We intend to provide an easily importable module that facilitates gamification integration and serves as an intermediary between Blackboard Learn and Unity games via REST API. This distinction enables a more personalized and direct interaction between Blackboard Learn and Unity 3D, improving overall developer convenience and use.

Overall, this project distinguishes itself from other solutions by offering a targeted and comprehensive integration of Blackboard Learn with Unity 3D. The package simplifies the integration process by automating the authorization and connecting of endpoints, allowing developers to construct interesting game experiences within the Blackboard Learn ecosystem. Developers can benefit from a more straightforward and user-friendly method of creating gamified learning experiences that connect smoothly with Blackboard Learn with this project.

3. DESIGN CONCEPT

3.1 Alternative Solutions/Approaches/Technologies

This part will cover several alternative solutions for gamifying Blackboard LMS.

One alternative solution is the use of game engines that have more functionality focused on educational gaming. Unreal Engine, for example, provides an extensive and flexible platform for developing high-quality educational games with enhanced graphics and physics simulations. Unreal Engine is also used to create virtual reality experiences, architectural visualizations, and training simulations. Unreal Engine provides a set of functions that lets developers request and retrieve HTTP requests. It also supports both synchronous and asynchronous requests. To use HTTP requests developers can use the FHttpModule class and access all the methods such as creating HTTP requests, handling responses, and setting headers. Because of its robust graphics rendering capabilities and adaptable toolset, it is a popular choice for many applications.

Another alternative for Unity could be the Godot game engine. Godot is an open-source and free game engine that was released in 2014. It was developed by a group of volunteers and is available for macOS, Linux, Windows, and other platforms. It supports C++, C#, and Python-like scripting languages. The engine also includes a powerful 3D and 2D physics engine that allows developers to create realistic simulations, games, VR, and AR. HTTPClient class allows developers to easily incorporate web-based functionalities into the games. Also, by using this class developers can handle errors and timeouts easily and monitor the progress of requests.

Instead of BB, alternative LMS such as Moodle, Canvas, or MindFlash could be considered, resulting in a more comprehensive and versatile solution that supports multiple LMS systems and expands the user base that can access the importable Unity package.

Furthermore, including advanced features in the Unity package can improve educational value even further - adaptive learning powered by AI (Artificial Intelligence) can provide students with tailored learning pathways, while learning analytics provide improved insight into student engagement and performance.

Moreover, in addition to UWP (Universal Windows Platform), Unity offers several other building types for different platforms.

1. Standalone: It is a default building type for Unity. It allows developers to build and distribute games for desktop platforms such as Windows, Mac, and Linux. The standalone build is suitable for creating games that do not require any specific platform-dependent features.
2. Mobile: Unity allows developers to build games for mobile platforms such as Android and iOS (iPhone Operating System). This build type is optimized for mobile devices and includes features such as touch input, gyroscope, and accelerometer.
3. Web: Unity has built-in support for building games that run-in web browsers such as Chrome, Firefox, Microsoft Edge, and Safari. This building type uses WebGL technology and can be played directly in the browser without the need for any additional plugins.
4. VR: Unity also supports building games for VR (virtual reality) platforms such as Oculus Rift, HTC (High Tech Computer) Vive, and PlayStation VR. The VR build type includes specialized features such as stereoscopic rendering, head tracking, and motion controllers.
5. AR: Augmented Reality (AR) is an interactive experience that combines physical and digital objects. Unity offers building types for AR platforms like ARKit (Augmented Reality Kit) for Android and ARCode (Augmented Reality Core) for iOS.

In conclusion, investigating alternative solutions such as using other game engines, expanding integration to other LMS systems, incorporating advanced features, leveraging virtual and augmented reality technologies, and considering the project's social and environmental impact can help to further optimize the integration of these two systems and contribute to a more engaging, accessible, and sustainable future for education.

3.2 Detailed description of Solutions/Approaches/Technologies of choice

Evidently, the current project consists of two major software components: Unity and Blackboard LMS. This section will provide a detailed description and rationale for selecting Unity and Blackboard LMS as the primary components of this project. The project team aims to explain the benefits and implications of their choices to give a comprehensive understanding.

Unity is an incredible multi-purpose real-time development platform that is commonly used for various video games, simulations, visualizations, VR, AR, 2D, and 3D. Developed by Unity Technologies, this program is acclaimed for its user-friendliness along with quick iteration times, plentiful assets, and tools readily available, and an active support community. Its libraries such as UnityWebRequest can make REST API calls to act as a client when incorporating Blackboard LMS with Unity.

Furthermore, compared to other game development platforms, Unity is simple to use. For example, building a simple game in Unreal will require more memory, space, and time, and an individual should be experienced enough to put all these game principles into the game. Unity allows games to run more smoothly on user machines rather than on other machines. Because Unity is easier to use, simple games such as mobile games and indie games are generally produced on it. Handling errors, animations, scripting, constructing environments, and creating importable packets in Unity is simpler, and low-end computers can run the games created in this due to Unity's efficient system design.

When it comes to programming language preferences, Unity supports C# programming language and C and C++ for plugins; however, as stated in the official documentation by Unity – C# is the main language they support. Hence, every script used in the implementation phase will be implemented using C# programming language.

Among the many features that have made Unity popular among developers is their implementation of HTTPWebRequest, an object-oriented wrapper around the HTTP protocol, as well as its newer version UnityWebRequest, which enables communicating with web APIs or other websites easier. Both objects are included in the.NET framework and can be used to make requests in nearly any language. Their contribution enables easy retrieval of endpoints through REST API and mapping of Blackboard LMS objects into unity game objects.

A more extensive description of UnityWebRequest is requested to help readers better comprehend its benefits. UnityWebRequest is a Unity technology that allows developers to submit HTTP and HTTPS requests from their applications. It simplifies interaction with online APIs and other web services by supporting GET and POST queries, adding headers and parameters to requests, and handling responses in a variety of formats such as JSON and XML.

It is built within the Unity game engine, making it simple for developers to start sending requests without the need for other dependencies. Additionally, it performs smoothly so developers don't need to be concerned about slowdowns or performance difficulties. UnityWebRequest now supports asynchronous requests, which allow developers to run requests in the background without interfering with the main thread of their program.

In addition to UnityWebRequest, there is HTTPWebRequest, an older object-oriented wrapper around the HTTP protocol that is still a useful tool

for making web requests but has fewer features than UnityWebRequest. Both are part of the.NET framework, which enables developers to make requests in nearly any language.

Another aspect of Unity that contributes significantly is the use of packages, which are collections of files and resources bundled together for developers to effectively share or collaborate on projects; each time a new project is created on Unity, a package is automatically created containing all project files as well as scripts, animations, textures, scenes, and other assets needed for development purposes.

In addition, UWP was chosen as the building type. Unity allows creators to pick UWP as their target platform in the Build Settings box to build games in UWP format. This generates a Visual Studio project that can be turned into an app that uses Windows 10 capabilities such as Cortana voice commands, Live Tiles, Xbox Live, and the ability to run on many devices with varied screen sizes and resolutions. Furthermore, UWP games can be published via the Microsoft Store, which can help them get visibility and reach a larger audience.

Moving on to BB LMS, it is an extremely recognized e-learning platform employed by educational institutions globally to facilitate communication between students and faculty members; features include a course management system, gradebook, discussion forums, content sharing, collaborate ultra, announcements, and group which have become immensely popular recently due to its capability increase overall learning experience for students and faculty plus provide more flexibility when managing courses.

Because the project's goal is to create a gamified learning experience for students, selecting an appropriate LMS was critical; Blackboard Learn was chosen after careful consideration because it is popular among academic institutions (including ADA University) and offers extensive features; and since 2014, it supports REST API integrations, further confirming the choice. After examining all the reasons presented, the decision was made to choose Blackboard LMS and Unity as the best solutions for the project.

3.3 Engineering Standards

Unity package is a collection of scripts, textures, prefabs, materials, etc. that can be easily transferred and reused across multiple Unity projects. Unity packages enable developers to develop and distribute bespoke functionality or content that streamlines the development process by mitigating the investment of time and effort. Although the Unity package does not require the developers to adhere to concrete guidelines for package organization, it encourages them to follow a standardized package structure to aid in the organization and user-friendliness of their package for others. One of the suggested structures for a Unity package looks as follows:

```
YourPackageName/
├── Assets/
│   ├── YourPackageName/
│   │   ├── Scripts/
│   │   ├── Prefabs/
│   │   ├── Materials/
│   │   ├── Textures/
│   │   ├── Animations/
│   │   ├── Editor/
│   │   ├── Resources/
│   │   ├── StreamingAssets/
│   │   ├── Scenes/
│   ├── ThirdPartyLibraryName/
│   │   ├── Plugins/
│   │   ├── Scripts/
│   │   ├── Editor/
├── Documentation/
│   ├── YourPackageNameManual.pdf
│   ├──
YourPackageNameAPIReference.pdf
├── LICENSE
├── README.md
```

The package responsible for integrating Blackboard LMS's elements to Unity Game Engine using REST API consists of two scripts, "APIManager.cs" and "SerializableItems.cs" located under the "Scripts/" folder which is a part of "Assets/", thus following the above-mentioned package structure.

The scripts are compliant with the below-given software engineering standards:

- SRP (Single Responsibility Principle) - Each method in the script has a clear and singular purpose.
- Singleton pattern - The class features a private static instance variable that is checked for existence in the "Awake()" method, guaranteeing that only one instance of the class is produced over the application's lifetime.
- Dependency injection - Classes are used as dependencies in methods and are passed as return values, allowing parts of the application to apply the classes without having to create new instances.
- Encapsulation - The APIManager class encapsulates all the logic required to communicate with the API and provides a simple interface for other classes to use.
- Asynchronous programming - Uses async/await to perform network requests without blocking the main thread, thus ensuring the application is responsive while waiting for a network response.
- Serialization - Uses Newtonsoft.Json to serialize and deserialize JSON data to enable storage and transmission of the object states.
- Exception handling - Uses try-catch blocks to handle exceptions that might occur in HTTP request handling and JSON parsing.
- Naming conventions - The methods use PascalCase naming case naming convention.
- Code comments - Includes comments to explain the general functionality of each method.

3.4 Architecture, Model, Diagram description

Fig 1. represents the general architecture of the project.

The process consists of the following steps:

1. Developer imports the package that enables the integration between Blackboard LMS and Unity to the environment.
2. Developer designs the environment of the game where certain game objects represent the elements of Blackboard LMS.
3. Developer uses the methods in "APIManager.cs" script to enable the API requests to the elements of Blackboard LMS.

4. Upon running the game, the methods written by the developer to give functionality to the game objects call the methods in "APIManager.cs" which send requests to the endpoints of Blackboard LMS.
5. Once the request is received by a certain entity in Blackboard LMS, it sends a reply back to Unity with a JSON response.
6. Received JSON response is parsed in "APIManager.cs" that call attributes of requested Blackboard LMS elements using serializable classes in "Serializable.cs".
7. Requested data appears in the game's interface when the return value in the certain method of "APIManager.cs" is received.

3.5 Social and environmental impact

The integration of Blackboard LMS and Unity has the potential to make a substantial social and ecological impact. By enhancing the educational value of games, this venture can add to the improvement of instruction and learning.

In terms of social influence, the project can offer an engaging and interactive learning experience for students. It has been widely acknowledged that traditional methods of education, such as lectures and textbooks, are often not enough to fully engage and motivate students. Game-based learning has been proven to boost student motivation and retention of information. By integrating Blackboard LMS with Unity, educational games can be created that align with the curriculum and provide a more productive way to deliver educational content - leading to improved learning outcomes as well as increased engagement and motivation among students.

Additionally, this project permits students to learn in a more collaborative and social environment. Games that incorporate Blackboard LMS can be designed to incorporate social features such as chat, collaboration tools, and leaderboards - encouraging students to collaborate, share their knowledge and experiences, as well as create a sense of community within the learning environment. This also helps reduce feelings of isolation among students who are studying online or remotely.

In terms of environmental impact, the venture contributes to decreasing paper waste and carbon footprint associated with traditional methods of education; providing an interactive and engaging learning experience through games reduces the need for printed textbooks or materials. Additionally, with its integration with Blackboard LMS, educational games can be accessed online which eliminates physical transportation for both students and materials.

In addition, it has a positive effect on fostering sustainable practices as well as environmental consciousness - educational games can include themes related to sustainability which inform students about environmental issues thus contributing to developing an environmentally responsible society. However, it's important to note that social and environmental impact will depend on how it is used and implemented - it is crucial that the games created are aligned with the curriculum and provide valuable instruction; they also must be accessible regardless of socioeconomic status or other factors; likewise considering potential negative impacts from excessive screen time is essential in using them responsibly and in balance.

Overall, combining Blackboard LMS & Unity has tremendous potential when it comes to both social and ecological influences by providing an engaging and interactive learning experience reducing paper waste and carbon footprint while promoting sustainable practices and environmental consciousness – however, utilizing them responsibly is key so benefits are maximized while avoiding any negatives pitfalls.

4. IMPLEMENTATION

4.1 Software Design

This part of the document will cover the modular architecture of the package that integrated

the Unity game engine and Blackboard LMS. The architecture has the following main components:

- the integration package.
- demo game - a user interface
- the backend scripts to control the game itself.

The integration package: this module includes an API manager and Serializable classes which are responsible for integrating the Unity game engine with Blackboard LMS. The package involves two main scripts which are called “APIManager.cs” and “SerializableItems.cs”.

- APIManager.cs

This script provides for authentication and authorization of the user, making API requests to and handling responses from the different entities of Blackboard LMS such as courses, assignments, grades, etc.

Fig 2: BaseURL information

This piece of code contains the variables that store the values of the client ID, client secret, redirect URI, and base URL for the Blackboard API. These values are used in various API calls throughout the script.

Fig 3: Get the authorization code.

This method generates an authorization URL and opens it in a web browser window. Then the user is directed to this URL and access to the Blackboard API.

Fig 4: Get a token.

This method retrieves authentication tokens by accessing an API endpoint and uses the "UnityWebRequest" object to send the HTTP POST request to the API endpoint. Once the request is complete, the method checks if the request was successful, and if not, logs an error message.

Fig 5: Refresh token

The piece of code is a part of the authentication process. It enables a user to refresh the access token.

Fig 6: Get course IDs.

This code is a method that retrieves the list of course IDs from a REST API. It constructs a GET request URL and sends it to the Learn API using UnityWebRequest class. Overall, it fetches IDs for a user from the Learn API, access token, and returns them as a list of strings.

Fig 7: Get course content by ID

This method sends a GET request to a specific endpoint URL with the course ID as a parameter. Once the operation is completed, the method checks the result and logs an error message if it is not successful. Finally, the method returns the courseContent object.

Fig 8: Get content's children contents by courseId

The method sends an HTTP GET request to a specific API endpoint using UnityWebRequest. It includes courseId and contentId parameters to get the children of course and content. If an exception occurs, the method logs the exception message using Unity's Debug.Log method. The method then returns the list of Content objects.

Fig 9: Get content’s attachments.

This method called GetContentAttachmentsAsync takes three arguments: courseId, contentId, and access_token. The method then loops through each file and assigns it to the lecture variable. Once the operation is complete, the method checks if the result was successful. If an exception happens, the error message is printed to the console. Finally, the method returns the lecture object.

Fig 10: Download the attachment.

This method downloads an attachment file associated with a specific course content. It uses UnityWebRequest to send a GET request to the

Blackboard Learn API and access token. The method then waits for the operation to complete if the operation is not successful, the method logs an error message. Then save the file on the user's computer using System.IO.File.WriteAllBytes(). If an error occurs during file writing, the method logs an error message.

- **Serializable.cs**

This script defines different entities of Blackboard LMS with their attributes i.e., Course entity has attributes such as id, name, ultraStatus, etc. The classes in "Serializable.cs" are used to define the structure of data received from API via methods written in "APIManager.cs" and provide the means for deserializing it into C# objects.

User interface: This module is responsible for providing the user interface for the game demo including design templates and implementation code. The user interface is designed to provide students with a gamified environment for their online courses. Upon entering the environment, students are presented with a school entrance scene. In the center of the school is a sphere that redirects students to the Blackboard login page where they can enter their login credentials to access their courses. Once logged in, students are returned to the game interface.

The courses are gamified with portals that, when approached, transport the student to the class entrance scene. In the class, there is a big board with three main sections. The left section displays the student announcements, which are posted by the instructors and include descriptions of course-related events or updates. The middle section displays the student's assignments, which may include instructions or file names. Clicking on an assignment automatically downloads it for the student. The right section displays the student's grades, including descriptions of the assignments, the grades earned, and the weighted total.

Finally, there are shelves that represent each week's materials. Clicking on a shelf displays the materials for that week, and clicking on a specific material automatically downloads it. If there is no material for a particular week, the shelf cannot be opened.

Overall, this user interface is designed to provide students with a visually engaging and intuitive way to access their online courses and course-related materials. Students are more likely to engage with their coursework and remain motivated throughout the semester by adding gamification features and sensible design.

The backend scripts: This module is responsible for integrating the backend of the game. This module includes assignments, environment, gamemanager, lectures, player controller, sound, soundmanager, weekdoors scripts.

Environment.cs: It is a script for controlling the game objects in the environment based on user input. There are three rings that enable, and disable the free-look camera, and store references to various game objects in the environment. The "portals", "weekShelves", "groups", and "grades" fields in the environment are arrays of transforms. The script has an "Awake" method that checks whether there is already an instance of the script and destroys any new instance. The "Start" method sets a boolean value to true. The "Update" method is called once per frame, rotates the rings, and checks for user input to enable or disable the free-look camera. Overall, this script provides basic functionality for controlling game objects in the environment based on user input.

Assignments.cs: It is a script that controls the downloading of homework in the game. When the user clicks on lecture, the "OnPointerClick" method is called in the game and then calls the Download Assignment. Overall, this script is designed to allow the user to download and view assignment content and attachments within a Unity game.

GameManager.cs: GameManager controls handling and refreshing the authentication and manages course content and assignments. Because the GameManager is a singleton class, there can only be one instance of it running the entire time the game is being played. This script has several methods to perform significant game actions. By using these

methods, it is possible to write text to a file, other than that, to leave room in the game, etc. Additionally, announcements, courses, and assignments, for instance, can all be stored in fields that are used to present information to the player. Moreover, there are sections for storing the text output components and the player's start position, which are used to set the player's position and show text to the player, respectively.

Lectures.cs: The lectures.cs script uses API manager and game manager instances to download a lecture's attachments when the lecture is clicked on in the game. When the lecture is clicked on, the method which is called OnPointerClick gets called. The DownloadLecture method looks for a lecture whose title matches the title of the clicked lecture. If the attachment is not null, it calls the DownloadAttachmentAsync method from the APIManager class to download the attachment.
PlayerController.cs: PlayerController as it is obvious from its name controls the actions and inputs of the player, like what happens when a game object is clicked by the player. The script includes methods for player movement, player login, and collision detection. In this script, the awake method checks if there is an existing instance of the script and destroys it. Other than that, when the Update method is called, it calls the PlayerMovement method and sets the rotation of a TextMeshProUGUI object to always face the camera.

Sound.cs and SoundManager.cs; SoundManager plays and stops the sound effects of the game. It manages audio playback in a Unity game. Awake () function checks if the instance variable is null or not. If there is already an instance of the SoundManager, the new one is removed. The Start () function is called after Awake (), and it is used to play two specific audio clips. Additionally, the Play () and Stop () functions are public methods that take in the name of the audio clip to play or stop. If the Sound object is not found, the function returns without playing or stopping any audio.

Overall, this script creates a simple audio manager that allows easy playback and stopping of audio clips in a Unity game.

Weekdoors.cs: WeekDoors script handles opening and closing the week folders if there is the content inside those files. This class implements the IPointerClickHandler interface, which allows it to handle click events on the door object. The OnPointerClick() function is called when the door is clicked, and it calls the OpenDoor() function. Before opening the door, the OpenDoor() function verifies that it is closed and that the text on the door's child TextMeshProUGUI component is not empty. When both requirements are true, the door is opened by using the DOTween library.

4.2 Essential Components of the Project

To demonstrate the functionalities the integration package offers to the developers, a demo game was built. The package for the demo game is different from the integration package and involves components other than scripts, specifically:

- Models

The Models folder is used for storing 3D models, which are assets that can be imported and used in the Unity environment. File formats such as FBX, and OBJ are stored here. In the demo game, this folder includes a 3D model for the main character (student) and his animations (walking).

- Materials

Materials folder stores materials, which are assets that define the surface appearance of 3D models in the scene. Assets in this folder are stored as “. mat” files that store information on the material's characteristics, such as the texture, colors, transparency, and so on. In the demo game, this folder includes materials for defining the appearance of doors in the classroom, rings, and ground in the school entrance scene.

- Plugins

The plugins folder contains third-party plugins and libraries that are used to extend the functionalities of Unity. When creating the demo game four plug-ins were imported to the package: DOTween_1_2_705, Epic Toon FX, JMO Assets, and TextMesh Pro.

- DOTween is a popular plug-in used for creating smooth and complex animations for game objects.
- Epic Toon FX is used for creating distinctive, eye-catching visual effects in the game such as a portal for the course spinning when the student enters it.
- JMO Assets plug-in is used by developers to quickly populate the game worlds with the library's pre-built 3D models, customizable materials, and textures. This plug-in was used in the game to generate more customization options for the visual representation of game objects.
- TextMesh Pro plug-in is a replacement for Unity's built-in TextMesh component that offers advanced typography and text styling capabilities for 2D, and 3D texts displayed in the game. Usage of this plug-in can be seen in the texts displayed on the assignment, announcement, and grades boards.

- Resources

Resources folder is used for loading assets at runtime, without explicitly referencing them in the code. Moreover, assignment files and course lectures downloaded by the student at runtime, are stored in this folder as well.

- Scenes

Scene folders include different scenes or levels that make up the game. Scenes in Unity are containers for game objects, assets, and other elements that make up the game world and are used for organizing and managing different parts of the game separately.

- Sounds

The sounds folder is used to store audio assets that can be used as music or sound effects in the game.

- Fonts

Font folders include a variety of font types such as TrueType, OpenType, and bitmap fonts that allow to create of custom text styles and improve the visual appearance of the game's interface. This folder in the demo game package includes OpenType font that can be seen on the texts above portals to the course and the student to display course names and student usernames, respectively.

- Scripts

The scripts folder is used to store all C# scripts that are used to define the behavior and functionality of the game objects used in the game. This folder in the demo game package includes scripts for controlling the player's (student) movements, portals, rings in the school entrance scene, etc.

4.3 Gantt chart

Fig.11 displays a complete timeline that the project team followed while developing the demo game and package.

4.4 Testing/Verification/Validation of results

During the testing phase of the project, the focus was to retrieve data from Blackboard LMS into Unity and map these data with the game objects and make sure that package was functioning correctly. It includes a sequence of tests that were designed to confirm the package's ability to communicate with Blackboard LMS via REST APIs. The first phase of testing included fetching data from endpoints using admin access, to verify the accessibility and wholeness of the JSON response. Authentication in BB LMS is role-based, meaning that the tokens are generated based on the role assigned to the users. The highest privilege belongs to the admin user followed by the

instructor and then the student. APIs tested during the first phase:

- **GET**
- /learn/api/public/v1/oauth2/authorizationcode
- **POST**
- /learn/api/public/v1/oauth2/token
- **GET**
  /learn/api/public/v1/courses/{courseId}/announcements
- **GET**
- /learn/api/public/v3/courses/{courseId}
- **GET**
- /learn/api/public/v1/courses/{courseId}/meetings/users/{userId}

The second phase of testing included verifying the access to endpoints with the student token. Since the package was developed targeting developers with a mission to gamify learning for students, it was mandatory to consider the access rights of students to BB LMS. To access the BB elements as students, redirection to BB LMS was built giving the users the ability to write their credentials to get authorization tokens from BB as students with limited rights for change. This was also mandatory to personalize the learning experience since courses taken in a term vary for each student. In the early stages, users had to add the authorization code to the game by themselves which was not a user-friendly approach. Then this was changed by applying deep linking. In the final version of the package, it was successfully tested, and the user was redirected into the game without adding anything extra other than a username and password. The only obstacle related to authorization was discovered when building the game since redirection to a third-party website is only available if the game in Unity is built using the Universal Windows Platform.

The next steps were to fetch courses, announcements, course content, and its children, grades, and groups with their attributes from Blackboard LMS. While no problems were faced with getting courses, course content, and grades, the announcement's body appeared in HTML format on the board object, not plain text. To resolve this problem, a method for converting HTML responses to plain text was developed. APIs used in this phase are listed below:

- **GET**
  /learn/api/public/v1/courses/{courseId}/contents
- **GET**
  /learn/api/public/v1/courses/{courseId}/contents/{contentId}/children
- **GET**
  /learn/api/public/v1/courses/{courseId}/contents/{contentId}/attachments
- **GET**
  /learn/api/public/v1/courses/{courseId}/contents/{contentId}/attachments/{attachmentId}/download
- **GET**
  /learn/api/public/v1/courses/{courseId}/contents/{contentId}/groups
- **GET**
  /learn/api/public/v2/courses/{courseId}/gradebook/columns/{columnId}
- **GET**
  /learn/api/public/v2/courses/{courseId}/gradebook/users/{userId}

The next steps involved testing the assessment's endpoint by getting the assessment and its questions. To fetch data from the assessment endpoint Blackboard had to be converted from its original view to Ultra. In this step, two types of assessment questions were tested: multiple-choice and open questions. The test was not successful because the JSON response from the assessment endpoint came differently for each user token. JSON response for admin token returns both questions and answer options in multiple choice questions. For instructors, JSON response comes with only questions, and for students, it comes neither with questions nor with answer options. In the case of open-ended types, questions can be retrieved with admin and instructor tokens, but again not for student tokens. It is also worth mentioning that in the current REST API

documentation of BlackBoard LMS, no APIs for posting the answers are available for the assessment's endpoint. APIs tested in this step are listed as follows:

- **GET** /learn/api/public/v1/courses/{courseId}/assessments/{assessmentId}/questions
- **GET** /learn/api/public/v1/courses/{courseId}/assessments/{assessmentId}/questions/{questionId}

Moreover, the assignments folder was tested to enable getting the instructions and attached files to it as well as submission by the student. Since assignments is not a default folder that comes with a course built by BlackBoard admins, but rather a folder created by instructors to structure the content of the course, the folder's ID is requested via API used to fetch course content. Then the assignment files inside the folder are taken via the API responsible for fetching content children. During the testing, it was discovered that a POST request responsible for sending a file was not available in the current documentation of BlackBoard REST API, so the submission of an assignment was concluded to be impossible for now. It is also important to mention that the assignments endpoint generated a JSON response without a body after the course was switched to Ultra view. To fix this problem a new course with the original BB view was created.

Other than the above-mentioned endpoints of the course, APIs for getting the user's name and avatar on the BlackBoard were tested. Also, a method was written to get the course content from "ROOT" folder; this folder gets created automatically when the course is copied into the BlackBoard and the get request was called using /learn/api/public/v1/courses/{courseId}/contents before checking for the folder with the name "ROOT". For the purposes of testing a POST request that was accessible to students, an API for posting an email from the BlackBoard was put into the test environment. APIs for the mentioned points are:

- **GET** /learn/api/public/v1/courses/{courseId}/users/{userId}
- **GET** /learn/api/public/v1/users/{userId}/avatar
- **POST** /learn/api/public/v1/courses/{courseId}/messages

Each test was conducted to verify that the importable unity package is fetching the necessary data from Blackboard LMS and accurately mapping it into Unity game objects.

Conclusion:

1.1. Discussion of results

The outcomes of our project illustrate the successful development and implementation of a Unity package that acts as a bridge between Blackboard Learn and Unity 3D. Developers can use the package to connect Blackboard Learn endpoints to game objects in the Unity environment, simplifying the integration process and improving the overall gamified learning experience.

In conclusion, the project has effectively achieved its primary objective of bridging the gap between traditional LMSs and Unity game mechanics. Throughout the development and testing phases, it was observed that the package effectively facilitated the transfer of data and interactions between Blackboard Learn and Unity 3D. By offering a new and engaging way for students to acquire knowledge, the project has successfully created a captivating learning experience. This personalized approach to education empowers students to learn at their own pace while giving developers an opportunity to create games and simulations that enhance learning outcomes.

Fortunately, the package was successful in its intended function; nonetheless, the obstacles and constraints encountered during the testing process must be addressed. The project encountered a number of challenges, including issues with

authorization and JSON answers from certain endpoints. Furthermore, the lack of documentation for certain APIs hampered the project's progress. Some developers stated a need for additional documentation and resources to help in the integration process, especially in higher-level scenarios. Therefore, it is important to consider these factors when evaluating the project's outcomes.

Ultimately, the Unity package provides a quick and easy way to integrate Blackboard Learn with Unity 3D, allowing developers to create captivating, gamified learning experiences. The positive feedback from developers, as well as the successful transfer of data and interactions across the two platforms, support the project's usefulness. Further improvements and refinements will be made in the future to address the identified limitations and ensure an even smoother integration process for developers wishing to create immersive and educational game environments within the Blackboard Learn ecosystem.

1.2. Future work

Although the created package succeeded in connecting the main endpoints of BlackBoard LMS with game objects in Unity, a number of improvements can be implemented both from the side of BlackBoard and developers/users of gamified learning. APIs for the main endpoints such as course content, assessments, gradebook, and announcements have already been put to the test and the observations were discussed in detail.

First, it is important to mention that the endpoints were tested for a BlackBoard course in the original view, except for the assessments since the API required the course to be in the Ultra view. However, assessment is not the only endpoint that requires the course to be in Ultra view; APIs for discussion boards remain not tested.

It was indicated that the current package integrates the main elements of the BlackBoard LMS for the students, more specifically, course content, assignments, assessments, announcements, and gradebook. However, there remain additional elements and functionalities of BlackBoard that need to be tested in the game environment; those include but are not limited to thecalendar, attendance, adaptive release, etc.

As stated above, to improve the current integration between Blackboard LMS and Unity, the limitations of REST APIs must be decreased. Restrictions such as fetching assessment questions being inaccessible with the student token or no API being available for the submission of a question's answer, or an assignment make it difficult for creating an interactive game to engage the students in learning. Without many POST requests being available to the students, the only function of the student character in the games will be to read or observe the instructor's actions without being to make an input which will make the game less interesting for the student. Other than posting in assessments and assignments endpoints being unavailable for students, the API also limits the interaction between the students who may be referred to as users of the potential games. One example of this can be observed while testing the groups element of Blackboard LMS as it is currently impossible to retrieve information other than the names of the groups the student is a member of; this means the student will not be able to see even the names of his/her groupmates in the game which restricts the chance of building a multi-player game in Unity.

In conclusion, the project has successfully created a Unity package that connects Blackboard Learn and Unity 3D; this provides a holistic integration experience, resulting in the enhancement of the overall learning experience for students. The project faced several challenges during the testing phase, including issues regarding authorization and insufficient documentation for specific APIs. Future work includes improving the REST APIs and testing additional elements and functionalities of Blackboard. This will ultimately allow for the creation of more interactive games and simulations that improve learning outcomes.

Appendices- Screenshots of the software interface

Figure 1: General architecture of Gamification 2.0 - Unity + Blackboard Learn project.

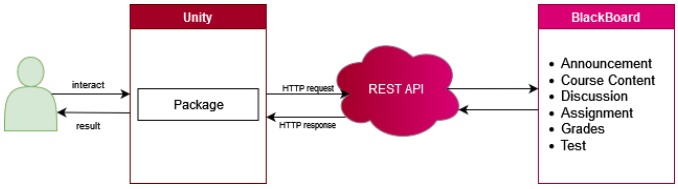


Figure2: BaseURL information

```
public class APIManager : MonoBehaviour
{
    public static APIManager Instance { get { return instance; } }
    private static APIManager instance;

    private const string CLIENT_ID = "fa5893df-6b24-49bd-aa0b-a37feabdeb0e";
    private const string CLIENT_SECRET = "eCMvFEVmJTjAXgI8ev34DefBW5RY0a4j";
    private const string REDIRECT_URI = "unitydl://";
    //private const string AUTHORIZATION_ENDPOINT = "https://ada-staging.blackboard.com/learn/api/public/v1/oauth2/authorizationcode";
    private const string BASE_URL = "https://ada-staging.blackboard.com";
```

Figure3: Getting authorization code

```
public void GetAuthorizationCode()
{
    // Redirect user to the authorization endpoint
    string authorizationUrl = BASE_URL + "/learn/api/public/v1/oauth2/authorizationcode?redirect_uri=" + REDIRECT_URI + SceneManager.GetActiveScene() + "&response_type=code&client_
    Application.OpenURL(authorizationUrl);
}
```

Figure4: Get token.

```
// Get token using authorization code
public async Task<AuthContext> GetToken(string authCode)
{
    AuthContext authContext = new AuthContext();
    var auth = System.Convert.ToBase64String(
        System.Text.Encoding.Default.GetBytes(
            $"{CLIENT_ID}:{CLIENT_SECRET}"
        )
    );
    var basicAuth = $"Basic {auth}";
    WWWForm form = new WWWForm();
    form.AddField("grant_type", "authorization_code");
    form.AddField("code", authCode);
    form.AddField("redirect_uri", REDIRECT_URI + SceneManager.GetActiveScene());
    using var www = UnityWebRequest.Post(BASE_URL + "/learn/api/public/v1/oauth2/token", form);
    www.SetRequestHeader("Authorization", basicAuth);
    www.SetRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }
```

Figure5: Get new token by refreshing

```
// Get new token using refresh token
public async Task<AuthContext> GetNewToken(string refreshToken)
{
    AuthContext authContext = new AuthContext();
    var auth = System.Convert.ToBase64String(
        System.Text.Encoding.Default.GetBytes(
            $"{CLIENT_ID}:{CLIENT_SECRET}"
        )
    );
    var basicAuth = $"Basic {auth}";
    WWWForm form = new WWWForm();
    form.AddField("grant_type", "refresh_token");
    form.AddField("refresh_token", refreshToken);
    form.AddField("redirect_uri", REDIRECT_URI + SceneManager.GetActiveScene()
    using var www = UnityWebRequest.Post(BASE_URL + "/learn/api/public/v1/oaut
    www.SetRequestHeader("Authorization", basicAuth);
    www.SetRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
```

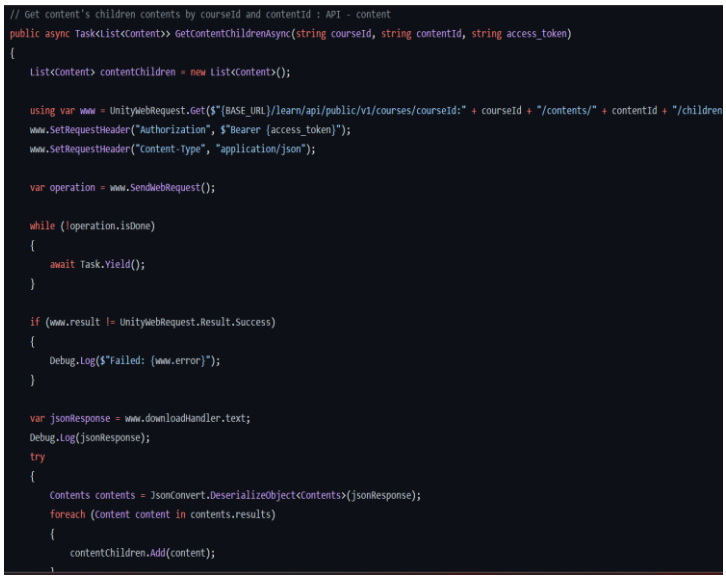
Figure6: Get course ID.

```
// Get registered course ids : API - course memberships
public async Task<List<string>> GetCourseIdsAsync(string user_uuid, string access_token)
{
    List<string> courseIds = new List<string>();

    using var www = UnityWebRequest.Get($"{BASE_URL}/learn/api/public/v1/users/uuid:" + user_uuid +
    www.SetRequestHeader("Authorization", $"Bearer {access_token}");
    www.SetRequestHeader("Content-Type", "application/json");
    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }

    var jsonResponse = www.downloadHandler.text;

    try
    {
        MyCourses myCourses = JsonConvert.DeserializeObject<MyCourses>(jsonResponse);

        foreach (Course course in myCourses.results)
        {
            courseIds.Add(course.courseId);
```

Figure 7: Get course content by ID

```
// Get courseContent by courseId : API - content
public async Task<Content> GetCourseContentAsync(string courseId, string access_tok
{
    Content courseContent = new Content();

    using var www = UnityWebRequest.Get($"{BASE_URL}/learn/api/public/v1/courses/co
    www.SetRequestHeader("Authorization", $"Bearer {access_token}");
    www.SetRequestHeader("Content-Type", "application/json");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }

    var jsonResponse = www.downloadHandler.text;
    Debug.Log(jsonResponse);
    try
    {
        Contents contents = JsonConvert.DeserializeObject<Contents>(jsonResponse);
        foreach (Content content in contents.results)
        {
            if (content.title == "Course Content")
```

Figure 8: Get content's children contents by courseId

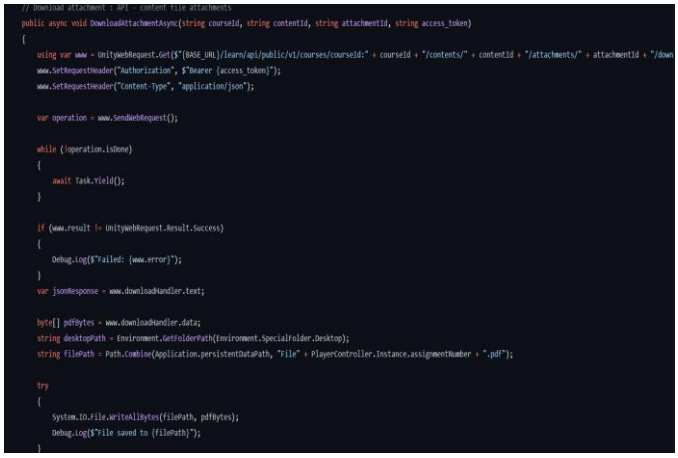


```
// Get content's children contents by courseId and contentId : API - content
public async Task<List<Content>> GetContentChildrenAsync(string courseId, string contentId, string access_token)
{
    List<Content> contentChildren = new List<Content>();

    using var www = UnityWebRequest.Get($"{BASE_URL}/learn/api/public/v1/courses/courseId:" + courseId + "/contents/" + contentId + "/children
    www.SetRequestHeader("Authorization", $"Bearer {access_token}");
    www.SetRequestHeader("Content-Type", "application/json");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }

    var jsonResponse = www.downloadHandler.text;
    Debug.Log(jsonResponse);
    try
    {
        Contents contents = JsonConvert.DeserializeObject<Contents>(jsonResponse);
        foreach (Content content in contents.results)
        {
            contentChildren.Add(content);
        }
```

Figure 9: Get content's attachments.

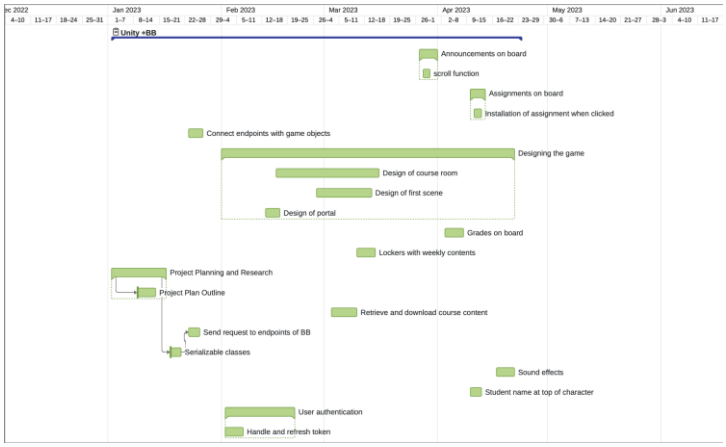
```
// Get content's attachments by courseId and contentId : API - content file attachments
public async Task<File> GetContentAttachmentsAsync(string courseId, string contentId, string access_token)
{
    File lecture = new File();

    using var www = UnityWebRequest.Get($"{BASE_URL}/learn/api/public/v1/courses/courseId:" + courseId + "/content
    www.SetRequestHeader("Authorization", $"Bearer {access_token}");
    www.SetRequestHeader("Content-Type", "application/json");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }
    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }
    var jsonResponse = www.downloadHandler.text;
    try
    {
        Files files = JsonConvert.DeserializeObject<Files>(jsonResponse);
        foreach (File file in files.results)
        {
            lecture = file;
        }
    }
    catch (Exception ex)
```

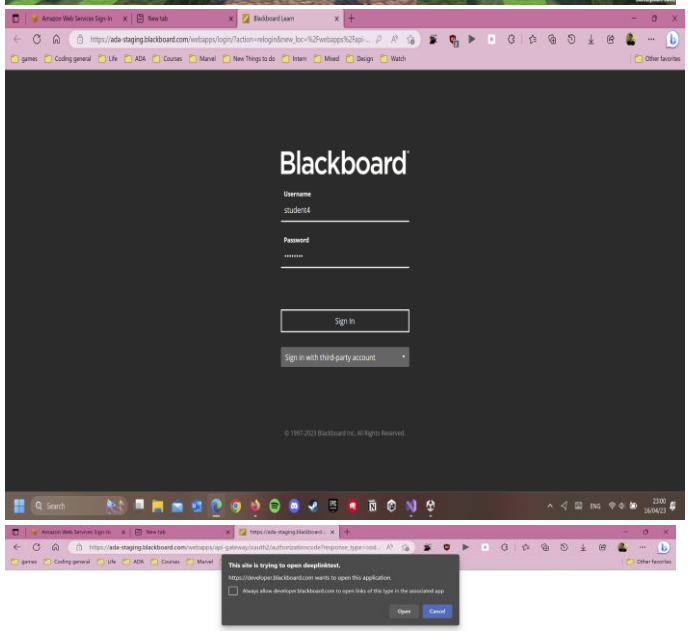
Figure 10: Download attachment

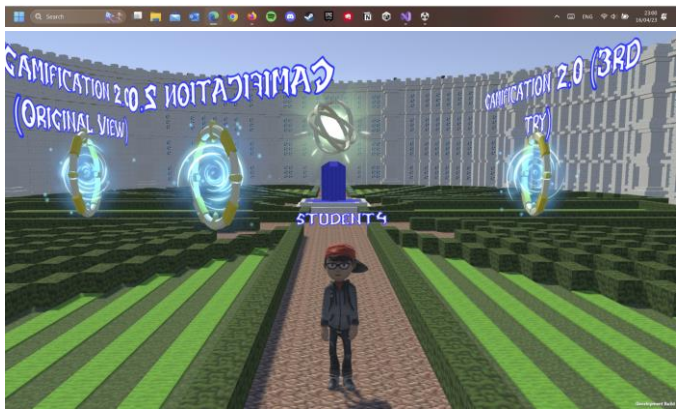


```
// Download attachment : API - content file attachments
public async void DownloadAttachmentAsync(string courseId, string contentId, string attachmentId, string access_token)
{
    using var www = UnityWebRequest.Get($"{BASE_URL}/learn/api/public/v1/courses/courseId:" + courseId + "/contents/" + contentId + "/attachments/" + attachmentId + "/down
    www.SetRequestHeader("Authorization", $"Bearer {access_token}");
    www.SetRequestHeader("Content-Type", "application/json");

    var operation = www.SendWebRequest();

    while (!operation.isDone)
    {
        await Task.Yield();
    }

    if (www.result != UnityWebRequest.Result.Success)
    {
        Debug.Log($"Failed: {www.error}");
    }
    var jsonResponse = www.downloadHandler.text;

    byte[] pdfBytes = www.downloadHandler.data;
    string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    string filePath = Path.Combine(Application.persistentDataPath, "File" + PlayerController.Instance.assignmentNumber + ".pdf");

    try
    {
        System.IO.File.WriteAllBytes(filePath, pdfBytes);
        Debug.Log($"File saved to {filePath}");
    }
```

Figure 11: Unity Package Development Gantt Chart for Blackboard Learn Integration







REFERENCES

[1] Bouchrika, I. (2022, September 26). Research. Retrieved from Research: https://research.com/education/best-lms-for-schools#1

[2] CHOU, Y.-K. (2015). Actionable Gamification. Octalysis Media.

[3] Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (pp. 9-15).

[4] Dicheva, D., Dichev, C., Gennady, A., & Angelova, G. (2015). Gamification in Education: A Systematic Mapping Study. Journal of Educational Technology & Society, 75-88.

[5] Healey, D. (2019, March 1). Springer Nature. Retrieved from Macmillan Education: https://www.macmillaneducation.es/wp-content/advantage/Gamification_White%20Paper_Mar%202019.pdf

[6] Kovács, D. (2014). Insights into the Cultural Heritage Landscape. Pécs: University of Pécs, Faculty of Adult Education and Human Resource Development.

[7] Livingstone, D., & Kemp, J. (2008). Integrating Web-Based and 3D Learning Environments: Second Life Meets Moodle. UPGRADE, 8-14.

[8] Machajewski, S. (2017). Gamification in Blackboard Learn. Blackboard World Conference 2017 (pp. 4-14). New Orleans: ERIC.

[9] Martínez-Ortiz, I., Antonio Calvo-Morata, A., Fernández-Manjón B. (2019). Introducing Competences into LTI-Connections between Learning Management Systems and Gaming Platforms. Dept. Software Engineering and Artificial Intelligence Universidad Complutense de Madrid.

[10] Maxwell, F. (2019). United We Stand: Platforms, Tools and Innovation With the Unity Game Engine. Social Media + Society.

[11] Neumann, A., Laranjeiro, N., & Bernardino, J. (2021). An Analysis of Public REST Web Service

APIs. IEEE Transactions on Services Computing., 957-970.

[12] Pishva, D., G, G. D., Nishantha, & Dang, H. (2010). A survey on how Blackboard is assisting educational institutions around the world and the future trends. 2010 The 12th International Conference on Advanced Communication Technology (ICACT), 1539-1543.

[13] Rahayu, F. N., & Ferdiana, R. S. (2022). Motivation and Engagement of Final-Year Students When Using E-learning: A Qualitative Study of Gamification in Pandemic Situation. Sustainability, 1-22.

[14] Rincon-Flores, E. G., & Santos-Guevara, B. N. (2021). Gamification during Covid-19: Promoting active learning and motivation in higher education. Australasian Journal of Educational Technology, 43-60.

[15] Richter, G., Raban , D. R., & Rafaeli, S. (2015, October). Studying gamification: The effect of rewards and incentives on motivation. Studying Gamification: The Effect of Rewards and Incentives on Motivation.

[16] Smiderle, R., Rigo, S. J., Marques, L. B., Peçanha de Miranda Coelho, J. A., & Jaques, P. A. (2020). The impact of gamification on students' learning, engagement and behavior based on their personality traits - smart learning environments. SpringerOpen.

[17] Uslucan, A., & Şenyer, N. (2013). SLOODLE: Usage as an Education Tool. AWERProcedia, 745-752.

[18] Winterhagen, M., Salman, M., Then, M., Wallenborn, B., Neuber, T., Heutelbeck, M., & ... Matthias. (2020). Introducing Competences into LTI-Connections between Learning Management Systems and Gaming Platforms. Journal of Information Technology Research.

[19] Wiegand, T., & Stieglitz, S. (2014). Serious fun-effects of gamification on knowledge exchange in enterprises. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), 321-332.

[20] Xu, Y. (2015). Effective Gamification Design: A Literature Review. The Standard International Journals, 47-54.

[21] Unity-SCORM-Integration-Kit. (n.d.). Retrieved April 30, 2023, from https://github.com/rstals/Unity-SCORM-Integration-Kit

[22] Zichermann , G., & Cunningham, C. (2011). Gamification by Design. Sebastopol: O'Reilly Media

[23] Bechtel, Michael, & Hart, Jonathan. "Unity + Bb Learn + Bb Data = Gaming + Education." Retrieved from Google Slides, : https://docs.google.com/presentation/d/1I3O8-hZRYImgcNiWkt7vHDclktLmJWdD/edit#slide=id.p1