Game-based Learning Tools for Educational Purposes with Virtual Reality Technology

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in Software Engineering

By
Thoai Ngoc Pham

May 2024
The thesis of Thoai Ngoc Pham is approved:

________________________________________________________________________
Professor Adam B Kaplan

________________________________________________________________________
Professor Vahe Karamian

________________________________________________________________________
Professor Li Liu

California State University Northridge
Acknowledgments

First, we would like to express our appreciation to Prof. Vahe Karamian and Prof. Adam B Kaplan. Thanks to their knowledge, experience, and counsel, we were able to overcome the challenges of this thesis during investigation and implementation. They were the ones to guide us through the thesis and show us our mistakes during the planning and implementation, as well as provide directions for our issues.

Additionally, we would like to send our gratitude to the instructors, professors, and staff at California State University, Northridge, especially the ones in the Computer Science department for educating new knowledge, consolidating our possessions, and supporting us throughout our time within the university. We appreciate the information that you have taught us so that we were able to wrap up this thesis.

Moreover, we would like to thank our friends who not only supported us physically through the time of this thesis but also mentally. Thank you for being there when we needed it.

Finally, we wish everyone the best. The time we have had here at California State University, Northridge will always remain one of the foremost memories in our minds. Once again, we would like to thank everyone for your support.
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>vi</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Why we chose this topic</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Main problems</td>
<td>1</td>
</tr>
<tr>
<td>1.3. Structure of the Thesis</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2. Related Work</td>
<td>3</td>
</tr>
<tr>
<td>2.1. Theoretical basis</td>
<td>3</td>
</tr>
<tr>
<td>2.1.1. XR study</td>
<td>3</td>
</tr>
<tr>
<td>2.1.2. Kahoot review</td>
<td>6</td>
</tr>
<tr>
<td>2.2. Technologies</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1. Unity Engine</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2. React Framework</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3. Clean Architecture</td>
<td>11</td>
</tr>
<tr>
<td>2.2.4. Networking</td>
<td>12</td>
</tr>
<tr>
<td>2.2.5. Database and Cloud</td>
<td>13</td>
</tr>
<tr>
<td>Chapter 3. Objectives</td>
<td>15</td>
</tr>
<tr>
<td>3.1. Goals</td>
<td>15</td>
</tr>
<tr>
<td>3.2. Requirements</td>
<td>15</td>
</tr>
<tr>
<td>3.2.1. Functional requirements</td>
<td>15</td>
</tr>
<tr>
<td>3.2.2. Non-functional requirements</td>
<td>16</td>
</tr>
<tr>
<td>Chapter 4. System Designs</td>
<td>17</td>
</tr>
<tr>
<td>4.1. Logical View</td>
<td>17</td>
</tr>
<tr>
<td>4.1.1. High-level design - Architecture</td>
<td>17</td>
</tr>
<tr>
<td>4.1.2. Class diagram</td>
<td>19</td>
</tr>
<tr>
<td>4.1.3. ERD (Entity Relationship Diagram)</td>
<td>22</td>
</tr>
<tr>
<td>4.2. Process View</td>
<td>24</td>
</tr>
<tr>
<td>4.3. Development View</td>
<td>25</td>
</tr>
<tr>
<td>4.3.1. Package Diagram</td>
<td>25</td>
</tr>
<tr>
<td>4.3.2. Network Diagram</td>
<td>27</td>
</tr>
<tr>
<td>4.4. Physical View</td>
<td>28</td>
</tr>
<tr>
<td>4.5. Scenario View</td>
<td>29</td>
</tr>
<tr>
<td>Chapter 5. Implementation</td>
<td>34</td>
</tr>
<tr>
<td>5.1. VR project implementation</td>
<td>34</td>
</tr>
<tr>
<td>5.2. Server implementation</td>
<td>40</td>
</tr>
</tbody>
</table>
Abstract

Game-based Learning Tools for Educational Purposes with Virtual Reality Technology

By
Thoai Ngoc Pham
Master of Science in Software Engineering

With the improvement in our quality of life nowadays, so do our entertainment needs. With the rise of applying the entertainment field to education, the effectiveness of expansion in terms of teaching is growing fast such as the language, STEM learning applications, etc. Besides the growth of the internet in many forms of protocol, there have been many ways to provide communication solutions between devices, including VR devices. However, to embed various latest technologies, there are not too many well-known projects that support edutainment. So in this thesis, the objective of this project is to implement an application with extensible tools to help the teaching environment. That also improves the user experience on the VR platform, and also a friendly website to manage user data and user customization for the tools. The first tool, which is going to be embedded into the virtual classroom for the demonstration of connectivity between lecturer and students, is implemented by taking the reference of the renowned game-based learning platform, Kahoot. The study will collect the potential of the VR environment in education.

Keywords: Virtual Reality (VR); Google Remote Procedure Call (gRPC); Game-Based Learning Tools (GLBT); Representational State Transfer (REST); Microsoft's Mixed Reality Toolkit (MRTK); Mixed Reality Toolkit (MRTK).
Chapter 1. Introduction

1.1. Why we chose this topic

In 2021, MRTK library, one of the best software development kits for mixed reality and augmented reality software applications won the Auggie Award for Best Developer Tool [2]. With the trend in VR applications in recent years, we challenge ourselves in this field with limited communication compared to other fields. After looking for some approaches in this field, our decision came up with the education and entertainment perspective. Moreover, with our experience on full-stack applications, 3 projects are built up for this thesis including server-side, VR client-side, and web client-side. Finally, there is a deployment step to accomplish the demonstration.

1.2. Main problems

In this project, we want to build a fully functional application with a mutual server in which users can join and study in the virtual classroom with each other and manage their customization for their tools if needed. However, for a better user experience, we want to see if adding a communication service like voice call can encourage users to stay engaged in our applications and improve our application overall. So we want to know:

- How do education and entertainment in the VR environment affect learning outcomes?
- How was the video streaming feature implemented from scratch?
- Can we develop a mutual server for VR and website clients?

Generally, these apps are assembled with multiple edutainment features including games, sharing, conversation, and extensible tools. We want to combine these features into our final project. This also raises the problem of resolving multiple user actions at the same time.

1.3. Structure of the Thesis

- Chapter 1: Introduction
  In this chapter, the project with its goals and contents will be mentioned.
- Chapter 2: Related work
  This chapter shows what we have prepared and researched before making this project.
- Chapter 3: Objectives
In this chapter, we will show what we need to complete this whole project.

- Chapter 4: System Designs
  We analyze the requirements for this project and from there we design diagrams following the 4+1 architecture.

- Chapter 5: Implementation
  The content of this chapter is how we implement the projects.

- Chapter 6: Deployment
  This chapter explains the deployment services.

- Chapter 7: Conclusion
  This is the conclusion on what we have accomplished on the project, its result, and future development plan.
Chapter 2. Related Work

2.1. Theoretical basis

2.1.1. XR study

The project focuses on VR implementation. However, for future work and more advanced study, it will inspect into the umbrella term which is extended reality or XR. To understand the XR technology, we need to consider the perception from our senses whether what we perceive comes from the digital or the physical world. When a person wears a headset, he/she can feel present in a digital environment. The digital information that they perceive through the senses overpowers the reasoning of the real. At that moment, their body feels it as real. The immersive experience could be seen through the virtuality continuum, a theoretical framework that visualizes the differences between the various technologies. The extreme parts in a continuum are different but the adjacent parts are indistinguishable. From the below figure, we can observe that XR is an umbrella term for AR, MR, and VR with low to high immersive of the physical to virtual environment.

![Extended Reality (XR), Augmented Reality (AR), Mixed Reality (MR), Virtual Reality (VR)](image)

Figure 2.1. Representation of current XR technologies by the spectrum of immersion [16]

Initially, let's take a look into the simple aspect of the XR usage survey inside the classroom. Oleg made an observation learning with XR in physics teaching from high school students in the 2019/2020 academic year to resolve two questions with negative to positive answers on 26 students [19]: Did students enjoy XR in physics classes? and does XR help
students learn physics? They gathered the results from students' cognition by creating an observational manager that contains students’ ideas like natural objects, artificial objects, space, and items in the physics classroom.

<table>
<thead>
<tr>
<th>Table 1: Did you enjoy XR using in physics classes?</th>
<th></th>
<th>Table 2: Does XR help you learn physics?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Answers of students</strong></td>
<td><strong>Number of students</strong></td>
<td><strong>Answers of students</strong></td>
</tr>
<tr>
<td>Strongly negative</td>
<td>0</td>
<td>Strongly negative</td>
</tr>
<tr>
<td>Negative</td>
<td>1</td>
<td>Negative</td>
</tr>
<tr>
<td>Neither negative nor positive</td>
<td>6</td>
<td>Neither negative nor positive</td>
</tr>
<tr>
<td>Positive</td>
<td>8</td>
<td>Positive</td>
</tr>
<tr>
<td>Strongly positive</td>
<td>11</td>
<td>Strongly positive</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>26</strong></td>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

Figure 2.2. Students' opinion about observation learning of XR in physics teaching results [19]

For those questions, the negative response to the first question was related to the religious restrictions of the student due to the student’s creativity while creating objects for the observational manager, and the second question explained the clumsiness and inconvenience of the XR headset. As a result, the first question showed that 19 students enjoyed it, and 6 did not decide; the second gave that 20 loved it, and 4 did not decide. From the survey, using XR technologies to enhance learning piqued students' intense interest. In addition to making them excited, XR is enhancing their academic performance in physics classes.

In addition, another point of view is whether XR technologies can improve the outcomes in learning by evaluating the student’s interactivity, engagement, and collaboration. There is a questionnaire on three different groups of students, each group has 20 people [9]. They prepared 6 questions to estimate the experiment's effectiveness by creating lessons on XR from very poor to excellent in 5 tiers total as follows: Are the lectures in XR interesting and attractive? Could lectures on XR improve student’s motivation? Does the application of XR help to keep attention on lectures? Could lectures on XR improve learning achievements for students? Could lectures using XR have an impact on the problem-solving skills of students? and Could lectures on XR improve the success of students?
For a Very poor scale, none of the students have chosen. The excellent tier covers the highest percentage in the responses as follows 75.83%, 73%, and 73% of student group 1, group 2, and group 3 respectively. The 2nd highest score is the good scale graded 14%, 14%, and 16% of student group 1, group 2, and group 3 respectively. For scale average, it is graded 9%, 12.5%, and 10% of group 1, group 2, and group 3 students respectively. Some of the rest with a small amount of percentage are about poor scores. The study's findings demonstrate that students' opinions about XR applications in higher education are favorable, as evidenced by their excellent grades by calculating standard deviation on excellent, good, and average scales of 10.82, 5.60, and 5.92 respectively.

Not only in higher education but also XR technology presents several exciting prospects for primary and secondary education as stated by Joshua [14]. XR is an experience-based learning tool that can give abstract ideas and concepts satisfactory visual representations. In contrast, to the discussion by Yiqun Liu [6], the use of VR, or XR in common, technology in education encounters many factors and limitations in the practice and application of education.
VR products and Technologies: The high price of the facility, its portability, and the limited contextual interaction experience.

2.1.2. Kahoot review

The reason we made the Quizzes, a Kahoot game similar, for our first tool on this application for the demonstration is based on the result of Guelnet Pinna on students' perceptions of using Kahoot [10]. The students had a positive opinion of the practical application of Kahoot as a tool for learning concepts, and it is assumed that the teacher made significant contributions as well. Even though there were a lot of mistakes made by the students, the use of gamification in the classroom with this tool was thought to be beneficial for the students because it increased their motivation and participation. Overall, the use of Kahoot was regarded favorably and encouraged student participation in the classroom. Students are motivated to play the game because of its gamified structure and design, which incentivizes them to aim for victory. This method of instruction made learning more engaging and enjoyable.

2.2. Technologies

2.2.1. Unity Engine

2.2.1.1. MRTK

According to XR Bootcamp [18], we have some potential packages to implement XR features on the Unity engine, with some advantages and disadvantages. Those three packages were initially picked because of the support for the XR development (three platforms AR, VR, and MR).

1. UNITY XR SDK

   The XR Interaction Toolkit is a part of Unity's XR Plugin Architecture (XR Tech Stack). It consists of a set of components that let developers create quick and easy XR interactions. Moreover, it identifies the input events by using Unity's Input System. The advantages of this package are due to very good documentation, a big community from Unity, the ability to easily implement basic XR interactions, and scalability. In contrast, developers could find it difficult to create custom features from the source. Because Unity's Input System is used in this library, which is more difficult to learn. Besides, there are no UI and hand interaction samples.

2. XRTK
The XRTK is a fork project of Microsoft’s MRTK. Another way to pronounce the XRTK is the ‘Mixed Reality Toolkit’. The XRTK fork was a result of MRTK developers from Microsoft and non-Microsoft MRTK contributors. It is confidently on a very flexible, expandable framework with a lightweight package. However, it is may difficult for beginners to approach and is slower development and support from core contributors of the library.

3. MRTK

Mixed Reality Toolkit developed by Microsoft in 2016, is an open-source software development kit for the development of MR and AR applications. It consists of a collection of components to support and enhance the mixed reality developers' and designers' experience. MRTK-Unity repository is used in the Unity Engine, it is a Microsoft-driven project that gives a utility to quicken cross-platform MR app development. Here are a few of its capacities: The cross-platform input system for spatial interactions and UI. In-editor reenactment. Extensible framework that provides the ability to swap out core components. A wide range of devices: Microsoft HoloLens, Meta Quest, Android, and iOS. From its capabilities, it is very robust support from Microsoft with a big community. It provides excellent sample scenes inside the project. It is a modular SDK that allows users to only download components, they need to reduce size and complexity. By contrast, the usage of this SDK differs from standard Unity conventions from modularity, which may confuse developers. Furthermore, there are many settings required to be configured properly.

2.2.1.2. VContainer

1. What is Dependency Injection?

Dependency injection is a programming technique in which an object or function receives other objects or functions that it requires, as opposed to creating them internally [13]. Dependency injection aims to separate the concerns of constructing objects and using them, leading to loosely coupled programs. Two well-known frameworks support Unity Engine that we will consider, VContainer and Zenject with some comparisons captured by VContainer [5].

2. VContainer

VContainer is a dependency injection framework for Unity Game Engine.

3. Zenject

Zenject is also a dependency injection framework built specifically to target Unity 3D (however it can be used outside of Unity as well).
4. VContainer and Zenject comparison

![Graph showing interaction test cases for VContainer and Zenject](image)

**Figure 2.4.** Interaction test cases for VContainer and Zenject [5]

![Graph showing memory allocation comparison for VContainer and Zenject](image)

**Figure 2.5.** Memory allocation comparison for VContainer and Zenject [5]

VContainer has better performance. For VContainer, most parts of reflections and assertions are isolated to the Container's build stage. It also has an easy-to-read documentation. Besides, VContainer has carefully selected features and does not register data-oriented objects in the container or actively inject the View component. This prevents the DI declaration from becoming overly complex. With the disadvantage, Zenject is often used to inject into dynamic or data-centric objects, but this is complicated. In VContainer, injection of MonoBehaviour is
recommended over injection into MonoBehaviour. Moreover, Zenject looks up all GameObjects in reflection when the scene starts, but this operation is expensive; VContainer does not do this. However, Zenject contains Signal for center messaging pattern, and Memory Pool to manage objects, but VContainer does not support it. Zenject has built-in resource injection on Unity. And Zenject supports injection with an identifier for every type of object.

2.2.1.3. Addressables

The Unity assets management could be easily solved by the Addressable Asset System by ‘address’ [17]. It reduces the asset management overhead by making asset packing distribution easier. It uses asynchronous loading to enable loading from any part of the implementation. Compared to bundle assets, resource folders, and direct references, Addressables provide an easier approach to making the game more dynamic. In the Unity engine, an asset is considered as whatever content is used to create the game. It can be a prefab, texture, material, clip, animation, etc. When an asset is assigned as ‘addressable’, it can be accessed from any location whether the addressable is located locally or network. Besides the addressable key, the assets could be grouped by label with different ways to load.

By using the Addressables, it could handle the following common asset management matters iteration time: asset content can be referred to by its address super handy. There is no longer a need to change the code in terms of optimizations; dependency management: not only content at the address assigned but also returns all dependencies of that asset by the system. For instance, all meshes, shaders, animations, and so forth are loaded before the asset is fully loaded; memory management: the Addressables provide both load and unload assets. There is a powerful profile that assists in identifying possible memory issues; content packing: bundle packing could be done efficiently because the system maps complex dependency structures even in cases where assets are transferred or renamed. It is easy to prepare for supporting downloadable content (DLC) and smaller application sizes for both local and remote deployment.

2.2.2. React Framework

For website development, we found 3 popular developing frameworks and how well they perform against each other, provide a structure to help judge front-end JavaScript frameworks in general and how can we choose the best fit for the project. The three frameworks are React, Vue, and Angular. They are all highly popular JavaScript libraries and frameworks that help
developers build complex, reactive, and modern user interfaces for the web. Because the project aims to manipulate data. That means the performance and the framework size is the thing we need to base on when choosing the front-end framework.

Angular uses a real Document Object Model (DOM), therefore it’s best suited for single-page applications where content is updated from time to time. It makes the process of updating much slower and in case of losing the flow, it will take a lot of time to find out the issue. Thanks to the two-way data binding process, all the changes made in the Model are replicated in the views securely and efficiently. Due to the wide range of features available, the application is much heavier (approximately 500KB) in comparison to Vue and React which slows down the performance a little.

Contrary to Angular, React uses a virtual DOM that enhances the performance of all sizes of applications that need regular content updates. Single-direction data allows better control over the project. The disadvantage might be the need for developers to constantly upgrade their skills due to the constantly evolving nature of React. As React doesn’t provide a wide range of libraries, its size is much smaller than the size of Angular (approximately 100KB).

Vue also uses a virtual DOM, so the changes within a project are made without affecting the DOM properly. Vue possesses the smallest size of the three (approximately 80KB) which significantly speeds up its performance. The next thing we need to consider is the popularity of the 3 frameworks and how complex to learn the frameworks. According to Stack Overflow Developer Survey Results 2019, React is the most loved by developers (74,5%) followed by Vue.js (73,6%) and only then Angular.js (57,6%) [15]. Because of React’s popularity, finding input components and ready-to-use elements is extremely easy. They’re all just a Google or GitHub search away.

After the research, we feel that React would be the best fit for the website project since it has these capabilities. The ability to reusable, composable, and stateful components. In a browser, we need to regenerate the HTML views in the DOM. With React, we do not need to worry about how to reflect these changes, or even manage when to make changes to the browser; React will simply react to the state changes and automatically update the DOM when needed. ReactJS is SEO-friendly. React JS comes with a helpful developer toolset. Alongside React, we consider React Redux, redux-saga, and Threejs to support the development. React Redux is the React UI bindings layer for Redux. It allows actions to be dispatched to the Redux store to
update the state and allows React components to read data from the store. With the Redux-saga library, it could be easy to manage, more efficient to execute, easy to test, and better at handling failures. For instance, asynchronous tasks like data fetching and impure tasks like accessing the browser cache. Three.js is a 3D library that simplifies the possibility of displaying a 3D content process on a webpage.

2.2.3. Clean Architecture

In 2012, Clean Architecture refers to an application organization style that has been around for nearly two decades [7]. The clean architecture is taken action from integrating some of the architecture of systems, including Hexagonal Architecture (a.k.a. Ports and Adapters) by Alistair Cockburn; Onion Architecture by Jeffrey Palermo; Data, context, and interaction (DCI) from James Coplien; Boundary Control Entity by Ivar Jacobson. They produce systems that are Independent of Frameworks which is independent of the presence of software libraries; Testable allows each component to be isolated for internal experiments; Independent of UI making UI easily changed without affecting the others; Independent of Database allows swapping out database management systems; Independent of any external agency simply isolates the system without outer world, respectively. From merging those architectures and their objectives by dividing the software into layers into a single practical concept.
Figure 2.6. Clean architecture [7]

From the figure above, the concentric circles stand for various software categories. Generally speaking, the software gets more advanced the further it goes. The mechanisms are the outer circles. Policies are the inner circles.

The Dependency Rule is the main principle that keeps this architecture functioning. Source code dependencies are restricted to pointing inwards by this rule. There is nothing that anything in an inner circle can know about anything in an outer circle. Specifically, the code in an inner circle cannot mention the name of something that has been declared in an outer circle. Functions, classes, variables, and any other named software entity fall under this category.

According to Ardalis [1], Clean Architecture aims to make sure that high-level modules and the abstractions they are associated with reside in an assembly or project that is independent of low-level modules or details, given the principles of dependency inversion. The primary tenet of the architecture is that this project, often referred to as Core or possibly Domain, depends on the other projects in the solution rather than the other way around.

2.2.4. Networking

From our experience, we considered the automation and the performance of communication to choose the suitable framework for the Unity client and .NET server. Looking at the RPC protocol, we know that gRPC was created by Google and used for a single general-purpose RPC infrastructure with high performance. Message Pack is used instead of Protobuf.

2.2.4.1. Message Pack

The MessagePack serializer is extremely fast for C# [8]. It outperforms other C# serializers and is 10 times faster than MsgPack-Cli. Also included with MessagePack for C# is built-in support for the lightning-fast LZ4 compression algorithm. Performance is crucial, especially for applications such as data caches, games, distributed computing, and microservices.
2.2.4.2. MagicOnion

There is a modern RPC framework that also provides bi-directional real-time communications such as SignalR and Socket.io and RPC mechanisms for the .NET platform, MagicOnion [3]. It utilizes MessagePack for C# to serialize call arguments and return values. .NET primitives and other complex types can be serialized into MessagePack objects.

2.2.5. Database and Cloud

2.2.5.1. Postgres

PostgreSQL is an object-relational database management system (ORDBMS) [11]. In addition, it supports a large portion of the SQL standard and offers many modern features, such as complex queries, foreign keys, triggers, updatable views, transactional integrity, and multi-version concurrency control. Also, PostgreSQL users can add new data types, functions, operators, aggregate functions, index methods, and procedural languages in a variety of ways.

2.2.5.2. Redis
Redis (Remote Dictionary Server) is an open-source, in-memory key-value data store for use as a database, cache, message broker, and queue. According to Amazon Web Services (AWS), Redis now delivers sub-millisecond response times enabling millions of requests per second for real-time applications in Gaming, Ad-Tech, Financial Services, Healthcare, and IoT. Redis is a popular choice for caching, session management, gaming, leaderboards, real-time analytics, chat/messaging, media streaming, pub/sub applications, geospatial, and ride-hailing [12]. All Redis data resides in memory, in contrast to databases that store data on disk or SSDs. By eliminating the need to access disks, in-memory data stores such as Redis avoid seek time delays and can access data in microseconds.

2.2.5.3. Firebase Storage

Cloud Storage for Firebase is designed for developers who need to store and serve user-generated content, like images or videos [4]. It is based on quick and secure Google Cloud infrastructure. It is a robust, user-friendly, and reasonably priced object storage solution built for Google scale. Regardless of network quality, the Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps. Users can access the files using Google Cloud Storage APIs manage buckets and download URLs on the server with the Firebase Admin SDK.
Chapter 3. Objectives

3.1. Goals

To achieve the objective of this thesis, we will incorporate these projects as one system using advanced technologies: The unity project is meant to implement the VR application for virtual room and tool usage of the system; There are two .NET projects for different purposes depending on the client side. The first project handed the HTTP2 over gRPC framework and streaming hub for the Unity project. The second one applied the REST style with HTTP1 for the website's simple API communication; The website project contains a management system for users on their tool data customization.

Those projects are using the same data which allows the same authorization for both client applications. Even though the technologies used to implement each project for the whole system could be very different from each other, there are reasons for them to fulfill the requirements for each platform. Here are the tasks we need to accomplish before working on the development: Study the common sections between those projects such as data, coding, and scope; Look into streaming for sharing screen solutions; Research the latest technologies applied to different frameworks and platforms; Research the potential of the VR on education in lacking resources in this area; Study on rendering 3D format data for a website and Unity engine; Research the architecture of each kind of project based on different frameworks; Look into the network protocol to support those different platforms.

3.2. Requirements

3.2.1. Functional requirements

About the VR application With the authentication feature, we have Login/logout: Users can log in and log out of the application. And some limited bypassing of the authorization on non-functional requirements.

In the tool playground, we provide View Tool Description: Users should be able to inspect the tool description including teaser, title, description, and user record; Select Tool: Users can select the tool to play.

Straight to the main feature, classroom manipulation, there are some functions provided Join Classroom: Users can join the classroom by room ID and their name; Create Classroom: Users can create the classroom with capacity and password; View Classroom Users: Users
should be able to know who joins the classroom; Update Classroom Setting: Users should be able to update the capacity of the classroom; Share Screen: Users can share their current user interface from the camera.

Finally, a demonstration of the tool in the system, the Quizzes, supports for Join Quizzes Game: Users can join the quizzes room by Quizzes room ID; Create Quizzes Game: Users can create the quizzes room; View Quizzes Users: Users should be able to know who joins the Quizzes game; Play Quizzes Game: Users can play quizzes from their interaction; Start Quizzes: Users can start quizzes game; Control Quizzes Stage: Users can update the quizzes stage including the next question, or end quizzes.

For the website, the system provided the same authentication with the VR application which is Login/logout: Users can log in and log out of the website. Besides, the website also provides the ability to manage the user data, including View Quiz Collections: Users should be able to inspect the quiz collections; Add Quiz: Users can add new quizzes; Update Quiz: Users can update quizzes to their collection; Remove Quiz: Users can remove quizzes from their collection.

3.2.2. Non-functional requirements

With some constraints based on the system functionality, we grouped the non-functional requirements as follows: Only logged-in users can create a classroom; Only logged-in users can manipulate classroom settings; Only logged-in users can share their user interface; Users should able to update their avatar and name after they join the classroom; All types of users should be able to interact with the shared screen syncing within 1 second; For quizzes, only students, who joined the classroom not created, can play the game; For quizzes, only users who created the classroom can manipulate the game stage.
Chapter 4. System Designs

4.1. Logical View

4.1.1. High-level design - Architecture

To have a better overview of how the system performs. The whole system will be covered in three sections following the clean architecture standard:

Presentation: this layer represents the client perspective, including the devices, applications, and services that connect to the other components.

Server: in the server side section, the gateway layer is responsible for the forefront layer to the client side. The business or use cases handling layer will calculate the request and communicate with the data services in the persistent layer. The persistent layer is the final layer in the server that controls the data from the database layer.

Database: the application used two separate database types to manage the data. One for cache management and another for permanent storage are Redis and PostgreSQL respectively.
Figure 4.1. Architecture Diagram
4.1.2. Class diagram

For the details of each main service used in the system, five diagrams are illustrated consisting of authentication, generic application, user services, and classroom, quiz hubs.

The authentication service is mainly for authenticating and authorizing users using the Identity API of .NET. There is an interface called IAuthService which shares accessible methods, Login, Register, and RefreshToken, to handle the authentication service. Currently, the application uses PasswordService, which implements that interface, for the authentication via user name and password. This service could be extensible for other services from third parties such as Google, Facebook, etc. It uses user data service for verifying users and JWT factory for accessibility management.

Figure 4.2. Authentication Service Class Diagram

For the generic application service, handles client version verification, server time sync, returning definitions, and environment configuration by using definition data and meta service. The metadata is stored in the database and can be edited by administrators. However, the definition data is controlled by developers on JSON files.
Figure 4.3. Generic Application Service Class Diagram

The user service is open for manipulating user data purposes. Only sync user data is used by RPC from the client at this time. User data is stored in the cache database and queried by a token from the request header.

Figure 4.4. User Service Class Diagram

The next diagram is about the classroom hub, which is over the socket protocol. The classroom hub implements join, leave, update users in the room, sync data, and invite to the multiplayer tool. Some sharable methods for getting user identity are inside of the GenericIdentityHub class.
The final class diagram is about the quiz hub. It handles all the real-time networking of the Quizzes tool including join, leave, get quiz collections, and gameplay manipulation. The same with the classroom hub, it also inherits the identity hub class.
4.1.3. ERD (Entity Relationship Diagram)

The diagram below shows the relationship between the entities in this system. Every table consists of createdAt and updatedAt time to keep track of each data. The user table uses two fields as the primary key, the EId or entity ID and identity ID, which points to the default identity model table. We used another table, the refresh token table, for user sessions to keep track of them and allow the system to increase security by managing short-term user accessibility.

The Quizzes tool utilizes the other two tables depending on the users, the quiz collection table and the quiz table. The relationship between them explains that a user may contain multiple quiz collections, and a quiz collection may have various quizzes. Those quiz data will be manipulated by users on the website, and it will be used in the Unity application.
Figure 4.7. ERD Diagram
4.2. Process View

From the process perspective, the activity diagram is used to illustrate the main gameplay for the whole system. When users start the application, they will see the landing screen. From the screen, the users can create a virtual classroom, join the virtual classroom with the existing room ID, open the tool description, or open the setting. Only authorized users can create the room, and start the multiplayer tool. After creating or joining the room, users can see the room status, and the virtual classroom with the ability to see others and edit their display name, and avatar.

![Activity Diagram](image)

Figure 4.8. Activity Diagram

24
4.3. Development View

4.3.1. Package Diagram

For a deeper overview of the architecture, we start with the package diagram on how system modules are distributed and interact. The rightmost container is about the Unity project, which could be reviewed in a separate section. There is a Shared component that is used in both the server and Unity project located at the deepest bottom. It is built of data transfer objects (DTO) which are physically stored in the Unity project, and interfaces or definition data models are stored in the server folder. The way the Unity project can understand the whole component is because of the autogenerated process to turn those objects and methods into other files according to a pre-configured location. The rest of the patterns including Core, Infrastructure, WebAPI, or GameRpc stayed on the server solution.

Figure 4.9. Overview of Package Diagram

First, the server consists of two applications, GameRpc, and WebAPI. They both utilize the other packages in the server solution. The description of each component in the server solution could be explained as follows.

GameRpc: the first application for RPC and Socket gateway, it mainly handles the GBLT or Unity client requests.

WebAPI: this application is for websites to manipulate data only on REST architecture.

Core: this project holds the most important components for the whole system, and its role is focused on the business logic.

Infrastructure: this project is responsible for database connective context and migrations.
Shared: this component contains the logic scope that could be shared between C# applications.

Figure 4.10. .NET Server Side Package Diagram

The Unity project might be simply split into three sections, user interface, services, and extensions.

User interface (UI): the design is managed by MVC architecture. The model component or data component could be only manipulated by the corresponding controller. The view or mono behavior attached to the suitable game object communicates with the controller to handle the functionality inside the designated game object. All in all, the whole MVC component for UI is managed by a Store service.

Services: this part hands on the crucial logic of the application. It handles the networking, application calculation, navigation, UI, and data management.

Extensions: it could be easily understood as a helper component. It contains the shared models from the server or some extended functionality to assist the development process.
4.3.2. Network Diagram

In the figure below, the left side is about the overall look of how networking works and the other side is about how the monitor streaming is performed.

In the first part of the application networking, some of the request types are needed the filtering on the client and server side for the authorizing process. However, all the RPC and bi-directional communications on the http2 protocol need to go through the serialization/deserialization step. Because those requests are mainly used for user interaction to get a faster response and better memory size. After bypassing those two processes, the server service will evaluate the request and response directly or broadcast.

The monitor streaming is almost built up from the host client, the one who creates the virtual classroom. From the host's perspective, the camera renders every needed layer into a texture renderer. Then, the texture renderer will be converted to the texture 2D, which is

Figure 4.11. Unity Client Side Package Diagram
compressed to the bytes and sent to the server. The server after receiving the image bytes broadcasts back to everyone in the room to render it on the monitor.

![Network Diagram](image)

**Figure 4.12. Network Diagram**

### 4.4. Physical View

The deployment diagram is used to illustrate the physical view of the system. From the devices, they access the artifacts in the local machine which could be a .apk file or azure web service to open the application. From the application type, the client will talk with different servers deployed by different virtual machines. The servers are using the same database whether permanent or temporary storage. For more information, we could see in chapter 6 deployment.
4.5. Scenario View

The scenario view will be discussed in three use case diagrams for distinct contexts, GBLT use case, Quizzes use case, and Quizzes archive use case. We can say the GBLT use case explains the core features of the system. Which serves the application flow allowing users to educate in the XR environment.
View Tool Description: from the landing page, any user can select the tool to see the
description including a trailer, content, title, etc.

Set Preferences: also from the landing page, they can set their settings.
Join Class: any user can join the class by inputting the room ID from the landing page.
View User List: after joining the classroom, users will able to see the others on a screen.
Create Class: only authorized users can create a classroom on the landing page.
Update Setting: only hosts who create a classroom can update the room settings when
they see the user list screen.
Select Tool: only hosts can select a tool from their subscription from the user list screen.
Share Screen: only hosts can share their screen with other monitors.

Figure 4.14. GBLT Use Case
The first tool introduced in this application, Quizzes, has its separate features from the main application listed as follows:

Join Room: any user can join the room by inputting the room ID from the landing page or getting invited from the host of the classroom. This room is different from the classroom, it is only served for the Quizzes tool requests.

View User List: after joining the room, users will able to see the others who joined the Quizzes inside the classroom on a screen.

Play Quizzes: only the player who is not a host can play the Quizzes after the host starts the Quizzes from the collection that the host chose.

View Score: once the question is due, the change of score view of that question will be shown up for all players.

View Score List: once the question is due, the host will see the score list of all players sorted by their score.

Create Room: only the host who creates the classroom can create the room for Quizzes by choosing the Quizzes tool on the landing screen.

Select Quizzes: only the host can select the collection they will use for this session.

Start Quizzes: only the host can start the Quizzes after they select the collection on the user list screen.

Start Next Question: only the host can manage to go to the next question or finish the game after the question ends.
The final use case tends to support the Quizzes tool. We call it Quizzes Archive which manages users’ Quizzes data such as quiz collection. It could be summarized as Quizzes
manipulation. Users can get the quiz collections list, add new quiz collections, and update, or remove the content of that collection including the quizzes or questions inside it.

Figure 4.16. Quizzes Archive Use Case
Chapter 5. Implementation

5.1. VR project implementation

The VR application is developed by using the Unity engine which follows the structure as figure below. The business is responsible for the interfaces, and they are implemented in the core section. The core will use the network and shared parts to handle everything consisting of event-driven, UI, service, game looping, and storage. There are some assistant parts during the development like editors or extensions.

![Figure 5.1. Unity Project Scripts Structure](image-url)
The final application flows as the images below. Almost all assets are used from the MRTK project. I just made some changes to the materials for the unique theme of our application. At first, users will see the welcome screen that follows the headset when they rotate.

Figure 5.2. GBLT Welcome Screen

By clicking on the start button, the landing screen will pop up, which offers the ability to join or create a classroom, log in or log out, close the panel, open the settings, or open the tools depending on whether the user is logged in or not based on device session cache.
Figure 5.3. GBLT Landing Screen for Unauthorized Users

Figure 5.4. GBLT Login Screen
After opening the tool, they will see the description including the video, contents, and user record as below. The play button at the bottom right only shows when they are host and in a classroom.
When users join a classroom, a classroom status screen will appear and give the teacher, and students information about avatars and display names. Moreover, they can edit their avatar, select a tool to play if they are the host, open classroom settings, close the panel, or quit the room.

Figure 5.7. GBLT Classroom Status Screen

Figure 5.8. GBLT Open Tool and Edit Player Data Panel
Here is the overview of the classroom from the developer view corner.

Figure 5.9. GBLT Overview from the Corner with Sharing Monitor

Coming to the Quizzes UI, we can see that there are two distinct perspectives, lecturer and student with different UI and UX. The lecturer manages the stage of the game, meanwhile, the students can only answer questions and wait for the score, rank, and next question.

Figure 5.10. Quizzes from the Lecturer’s Perspective
5.2. Server implementation

The server includes two applications as mentioned above on component diagrams including GameRpc, and WebAPI for different client applications purposes. Let's discuss the logical business on the server side, which are Core and Infrastructure projects. Inside the Infrastructure project, there is the data folder, which contains the database contexts for communicating with the actual database service, and the migration folder to keep track of the data model version. For the Core project, the interfaces for the whole server applications or internal services are implemented in this scope. Besides, it could be mentionable components, which are domain and specification configuration, help manage the data structure of tables and query the database more easily.
The below figure shows the dependency packages needed to build the GameRpc server. It could be grouped to solve dependency injection, relational database connection, RPC, real-time communication, authentication, and caching problems.

Figure 5.12. Server Structure

Figure 5.13. GameRpc Server Dependency
Similarly, the WebAPI server is quite similar to the GameRpc server above, but different connection that uses the MVC architecture.

![WebAPI Server Dependency](image1)

Figure 5.14. WebAPI Server Dependency

Finally, both server applications use the same settings for the endpoints using Kestrel configuration on Http1 and Http2 protocols.

![Kestrel Endpoints Configuration](image2)

Figure 5.15. Kestrel Endpoints Configuration for GameRpc and WebAPI servers

5.3. Website implementation

The website or dashboard page is another project in this system. It used the React framework to build up, which follows the structure in the figure below. The App script points to routing and then navigating to corresponding pages. From each page, they use different features implemented inside the extensible features folder by using the same store management. The assets of this website utilize the Kahoot website.
Figure 5.16. React website structure

When accessing the correct endpoints of the website, the welcome page is displayed below. Users can press the Join Now button to log in or register the account. Then, they will see the dashboard or home page.

Figure 5.17. Website Welcome Page
From the home page or navbar on the top, users can go to the Quizzes archive page to maintain their Quizzes data. They can edit, delete, or search the collection that they created.

From each quiz collection, users can add new questions, or edit the existing ones by clicking on the details edit dropdown from the edit button. They can update or delete the question, duration, answer options, image, or 3D model and save it on the top right button.
Figure 5.20. Website Quizzes Collection Edit Page

Figure 5.21. Website Quizzes Collection Edit Page with Model Upload
Chapter 6. Deployment

6.1. Unity Project Deployment

The Unity build will end up with .apk for the VR application. The file can be accessed in the website project. The following is capturing the memory usage to estimate the calculation based on the number of students inside a room. In the project, for simplicity and a high refresh rate, one student model including some body parts is summed up around 2 KB. In addition, the environment will be counted such as seat, table, etc., the total student object size leads to just under 700 KB including Unity native objects and managed objects in scripts.

![Figure 6.1. Memory Size of Student Object in the Profiler](image1)

![Figure 6.2. In-Project and Expected Memory Size of Student Object](image2)
In summary, the total size of one student object could be around 700 KB - 10 MB depending on how complicated the assets are being used on each model including mesh, texture, shader, material, etc. Eventually, memory does not matter due to of the nowadays device system configuration but the processor does. During some experiments when testing on the actual device, we found out that if the total memory size of a single student object is over 10 MB, that could lead to laggy, and it drops the FPS of the application to under 30. Around 6-7 MB per single object for the device processor to handle the rendering or draw call of each frame to get the most friendly user experience.

6.2. .NET Servers and Website Deployment

In this thesis, we use the free Azure services to deploy everything, especially two basic subscription virtual machines with limited memory and RAM. From the image below, we can see the public IP address, PostgreSQL, and virtual machine with its related such as network interface, network security group, and disk.

Figure 6.3. The Resource Group for Node and WebAPI Deployment

Besides Azure deployment, there is also the cache database deployed in the Redislabs provided by Redis and Cloud Storage using Firebase.

All of the services we mentioned above are using free subscriptions except for dynamic public IP addresses which is $0.004/hour. In summary, the total cost each month is around $5.76 including two public IP addresses for two distinct server programs.
Chapter 7. Conclusion

7.1. Accomplishment

In this project, we have been able to accomplish the following things: we have a chance to learn and understand the frameworks, and technologies during implementation. Found solutions to the potential problems mentioned earlier. Besides, we could complete the implementation of various network protocols. With full development and deployment process for different projects. Able to implement the features for learning assistance such as screen sharing, and extensible tools. Also, able to look into the potential of the VR platform in education through some existing surveys.

From the beginning, when the decision to use the Unity engine was made, I heavily considered the scalability and the distribution of the architecture for the whole system. That is why the .NET server and Unity engine use some similarities in generating or sharing assets. The tiny mismatch for React instead of Blazor using C# is because of our experience with React and the aiding of the 3D rendering module.

7.2. Limitations

While immersive reality technology can address different teaching methods, bringing about educational reform through the use of new technology is a long process. It is a long-term project to implement immersive virtual reality technology in the classroom, for distance learning, and other educational activities. For the technical side, the hardware might be costly and inconvenient compared to the other devices. Moreover, the interactive context supplies could be limited. On the educational side, there is a lack of educational resources and incompatibility between existing educational technologies.

7.3. Future work

The potential of this idea is a long-term decision. As an extensible tool for education, the outline of this project depends on the subject's ideas that could apply to the VR world such as geometry, physical demonstration, chemistry reaction, etc. The foremost works, that could be observed from the current state, would be listed in these items:

Upgrade the system to support in-person learning using MR and AR technology.
Improve the UI/UX for the VR application.
Add voice chat to support remote communication.
Add a user bundle tool to upload materials of the models on the webpage.
Add more rules and verification filters for security and permissions.
Implement the assistance AI for the whole system.
Cache more data for the read query of the database related to the user preferences and room state.
Enhance the capability of deployment artifacts.
Add history usage record to track the user's activities including anonymous.
Add more tools to support various educational areas.
References


Appendix: List of Figures

Figure 2.1. Representation of current XR technologies according to the spectrum of immersion
Figure 2.2. Students' opinion about observation learning of XR in physics teaching results
Figure 2.3. Students' answers to the questionnaire
Figure 2.4. Interaction test cases for VContainer and Zenject
Figure 2.5. Memory allocation comparison for VContainer and Zenject
Figure 2.6. Clean architecture
Figure 2.7. MessagePack, JSON, Protobuf, ZeroFormatter serialization comparisons
Figure 4.1. Architecture Diagram
Figure 4.2. Authentication Service Class Diagram
Figure 4.3. Generic Application Service Class Diagram
Figure 4.4. User Service Class Diagram
Figure 4.5. Classroom Hub Class Diagram
Figure 4.6. Quiz Hub Class Diagram
Figure 4.7. ERD Diagram
Figure 4.8. Activity Diagram
Figure 4.9. Overview of Package Diagram
Figure 4.10. .NET Server Side Package Diagram
Figure 4.11. Unity Client Side Package Diagram
Figure 4.12. Network Diagram
Figure 4.13. Deployment Diagram
Figure 4.14. GBLT Use Case
Figure 4.15. Quizzes Use Case
Figure 4.16. Quizzes Archive Use Case
Figure 5.1. Unity Project Scripts Structure
Figure 5.2. GBLT Welcome Screen
Figure 5.3. GBLT Landing Screen for Unauthorized Users
Figure 5.4. GBLT Login Screen
Figure 5.5. GBLT Landing Screen for Authorized Users
Figure 5.6. GBLT Tool Description Screen
Figure 5.7. GBLT Classroom Status Screen
Figure 5.8. GBLT Open Tool and Edit Player Data Panel
Figure 5.9. GBLT Overview From the Corner with Sharing Monitor
Figure 5.10. Quizzes from the Lecturer’s Perspective
Figure 5.11. Quizzes from the Student’s Perspective
Figure 5.12. Server Structure
Figure 5.13. GameRpc Server Dependency
Figure 5.14. WebAPI Server Dependency
Figure 5.15. Kestrel Endpoints Configuration for GameRpc and WebAPI servers
Figure 5.16. React website structure
Figure 5.17. Website Welcome Page
Figure 5.18. Website Home Page
Figure 5.19. Website Quizzes Archive Page
Figure 5.20. Website Quizzes Collection Edit Page
Figure 5.21. Website Quizzes Collection Edit Page with Model Upload
Figure 6.1. Memory Size of Student Object in the Profiler
Figure 6.2. In-Project and Expected Memory Size of Student Object
Figure 6.3. The Resource Group for Node and WebAPI Deployment
Figure 6.4. The Resource Group for GameRpc Server Deployment