Technological Feasibility

Grace Hsieh, Brett Lewerke, and Chayson Spigarelli

1 April 2022

Sponsor Signature

Team Lead Signature

Project Sponsor: Terry E Baxter

Faculty Mentor: Dr. Michael Leverington

Table of Contents

1.0 Introduction

Many students are required to submit work, track course progress, and view grades through an online tool provided by the university, called a learning management system (LMS). A LMS is a piece of software used for the administration, documentation, reporting, automation, and delivery of educational courses, training programs, or learning & development programs. Common features of a LMS include: a gradebook, calendar, class material, user account info, and their school information. Most learning management systems are outdated because they do not implement enough functionality. For example, the Blackboard Learning Management System (BbLearn) does not allow teachers to customize a course in a way it could be gamified without introducing extra maintenance. Moodle is another example of a LMS but one of the main disadvantages is that it's very difficult to set up and fine tune. We are looking to build a cost-effective solution for an LMS that is not difficult to set up and easily customizable.

Our client's name is Professor Terry E Baxter who studies Civil and Environmental Engineering at Northern Arizona University (NAU). BbLearn is the LMS at NAU which has much distaste throughout NAUs students and faculty members. Courses cannot be gamified without causing extra work for a faculty member. The current workflow of our client is tedious because he has a currency he needs to keep track of manually with a Microsoft Excel spreadsheet. BbLearn cannot track the currency on its own. We plan to build tools on our own LMS that help prevent valuable time being spent on keeping the gamified course up to date by our client.

Students today have a variety of distractions at their disposal making it more important than ever to keep students engaged. Gamifying a course could help students become more engaged because most people enjoy playing games rather than doing schoolwork. Engaged students will perform at a higher level in the class because they will spend more time on their studies. The goal of Team GamingEd is to increase student engagement with a LMS by designing an interactive platform that is enjoyable to use. The team wants to allow our sponsor to gamify their course with ease using a desktop application that our team will design. The team also wants to allow for students to take the course on a website application. The team's product will be better than our clients current gamified solution because it will automate currency exchanges and allow students (players) to choose avatars. The solution we have is to create a new LMS that allows for teachers to easily gamify any course and free up the restrictions of BbLearn. The technologies the team will use in order to develop the software are discussed in the different sections below.

In order to create a gamified LMS, Team GamingEd has identified the key design challenges, assessed potential technologies using our desired characteristics and custom rubric, before deciding on the technologies that will be used and integrating these into an architecture. In the Technological Challenges we identified the desktop application, website application, database, and API as the main design challenges. In the Technological Analysis section the team assessed

solutions such as Unity, Unreal Engine, and CryEngine in order to make a fantastic looking desktop application. The team also addressed solutions for designing the website application such as: .NET Frameworks, React, and not using a framework. The Technology Integration section discusses how the team chose solutions to different problems and how those chosen solutions will come together in order to create a gamified LMS. This section also introduces a system diagram of our envisioned architecture which shows how the major elements relate to each other.

2.0 Technology Challenges

There are four main design challenges that need to be considered for the successful development of the new gamified LMS which are outlined in this section. The chosen solutions provided to each challenge are the backbone and starting point for future development.

- **Desktop Application:** The desktop application allows the administrator to easily create, manage, and plan class content even while offline. These courses will then be uploaded for student use on the website application. Along with offline capabilities, the chosen solution needs to have built in interactivity and rendering functionality to create an interactive and intuitive GUI.

- **Website Application:** Students, or players, will be able to access and complete courses created and saved from the desktop application. The structure of the class and its contents will be determined by how the administrator designed it. The players will view and interact with the finished class to be able to submit assignments and exams. The administrator will choose to either manually grade assignments or create predetermined correct answers to be compared against.

- **Database:** All interactions recorded on the desktop and the website application will be stored to the database. Data will include course content, such as the structure of the class and all material needed to pass the levels, and individual user information, such as encrypted log-in credentials, any grades, and personal progress. The key features that need to be focused on is speed of allocation, security of the data, and upholding ACID industry standards. ACID stands for atomicity, consistency, isolation, and durability.

- **API:** The API will manage interactions and data between the database and both the desktop application and the website application. The API needs to be able to actively create new tables for information such as student grades, test information, and log-in information. The content of the data is based upon the structure of the class that the teacher builds.

3.0 Technology Analysis

Each design challenge will be detailed in a separate technology analysis. After understanding the necessary key characteristics of a challenge, several alternatives will be researched and scored based on those characteristics. A final decision will be determined on the score and plans will be made to create working proof of concepts to be used in a requirements specification document.

3.1 Desktop Application

The desktop application's main function is to allow administrators to create classes to their own specifications. Making the classes will also be gamified to provide clarity as to what a student can be expected to go through and ensure that the class creator understands the idea of gamifying. While looking for solutions, the team started with the theme of gaming and began looking into video game engines that satisfied the basic requirements for the project. The core characteristics of the engine are being interactive, having rendering capabilities, and being able to be used offline. Online functionality will be discussed in the API section of this document.

3.1.1 Desired Characteristics

Due to the nature of gaming engines, there are a lot of extra features with AAA games in mind. In other words, the main target audience for these products is large high-budget and high-profile companies with team sizes spanning from a dozen to several hundred. Therefore, when scoring each characteristic for a three person team it is easier to start at a number of points and give reductions for complications, or inconveniences that would cause the team to pursue additional hours of production, predominantly a lack of built-in systems (A) or overly complex built-in systems (B). Each category will start at five points and each reduction will remove a set number of points outlined in each characteristic of this section. With a total of five characteristics, the maximum possible score an alternative could achieve is 25.

- **Interactive:** There must be tools or scripts that allow for features such as buttons and other manipulatable objects for administrators to interact with in order to create and modify class designs as well as navigate through the program and all of its possible features, such as accessing grades and student information. Other actions that need to be functional include but are not limited to text input, drag-and-drop, and scrolling.
  Possible Complications:
  A) Interactive objects need to be made from scratch or are not built-in (-2)
  B) A learning curve that would take longer than two hours (-2)
  C) Text based technology (-5)

- **Rendering:** Viewing the content of the class requires 2D rendering in order to display the relative position of images that would represent levels in the class. For example, popular video games such as Candy Crush are designed with the intent of having users navigate and complete levels to unlock more levels of the game. For the LMS, each level will contain study material and exams that must be completed to progress through the class. Several interactable menus in the form of stages is ideal to manage the player's current view of the program.
  Possible Complications:
  A) Cannot display basic geometric shapes without additional work (-2)
  B) A learning curve that would take longer than two hours (-2)
  C) Not possible to display any geometric shape whatsoever (-5)

- **Offline Functionality:** Administrators must be able to build, save, and modify existing courses that have been locally saved to their computer. All necessary assets for doing so would also be available to them during this process. Any classes made in this way would be saved locally to be uploaded at a later time when internet access is available.
  Possible Complications:
  A) Requires additional setup to interact with a possible API (-2)
  B) A learning curve that would take longer than two hours (-2)
  C) Does not have offline capabilities (-5)

- **Programmer Ease of Use:** Due to the time constraints of the Capstone class, the team's current skill set and amount of available documentation are being factored as a necessary characteristic. The team wants to work with a program that is in a language that the team can easily learn or already understands. Detailed documentation will assist any learning curve necessary and save time when trying to find a solution to technical challenges found while developing the application.
  Possible Complications:
  A) i) One team member does not know the language (-1)
  ii) Two or three members do not know the language (-2)
  B) Documentation is limited, determined by the census of the internet (-2)

- **Cost:** Generally, game engines are marketed in two ways. Firstly, subscription services have different tiers ranging from free to several hundred dollars a month. Each increasing tier gives the buyer more features associated with the engine such as third-party support and built-in tools. Secondly, published projects that generate revenue may have to give some percentage of gross profit to the developers of the engine, but the amount and conditions that payment starts vary too wildly from company to company and need to be evaluated on a case by case basis. The team wants to avoid needing to ask for financial assistance as much as possible and would prefer a free version that does not require

immediate costs. Royalties have been considered a future problem at this time due to being unsure if the product will be sold for individual use or use by institutions as a whole, such as universities or educational platforms, and will be noted for future use.
Possible Complications:

A) Subscription required to fulfill other desired characteristics (-4)
B) Additional one time purchases required to fulfill other desired characteristics (-1)

3.1.2 Alternatives

- **Unity:** Created by Unity Technologies. The latest version of this gaming engine was created in 2017 and includes purchasable upgrade plans that include features such as source code access, additional technical support, and real time diagnostics. Indie games are predominantly created using this software. Some recent notable games include Hollow Knight, Cuphead, and Escape from Tarkov.

- **Unreal Engine:** Developed by Epic Games and Digital Extremes. Unreal 4, made in 2014, is the current usable version the team would have access to as of March 7th., 2022, Unreal 5 is projected to be released some time in 2022. A few games that use the engine include Unreal Tournament, Life is Strange, the Borderlands series, and Mass Effect.

- **CryEngine 3:** Created by Crytek. CryEngine 3 was released in 2018 with the most recent stable release being July 2020. Some more notable games made with this engine are Sniper Ghost Warrior 3, Hunt Showdown, and Warface.

3.1.3 Analysis

All alternatives met basic requirements listed in the desired characteristics section. Consequently, a choice was made based on the current skill set of the team and how difficult it would be to implement our solution using each software. If the engine is overly complex it would cost the team time to understand the basics of the program before beginning work. Similarly, if the engine is too simple then the team would have to invest time to implement more complicated functionality in order to meet the project requirements. Therefore, points were reduced in accordance with section 3.1.1. See Figure 1 for a full breakdown of the scores.

- **Unity:** The only reduction given to Unity was that one of the team members are not currently fluent in C#.
  Total Score: 24

- **Unreal Engine:** Unreal Engine has a mandatory learning curve due to the BluePrint Engine. The BluePrint Engine is a custom UI engine made to operate and navigate Unreal

Engine's large number of features. Although it is very detailed, the node based structure is confusing and foriegn to the team. To learn how to use it would be the same as learning an entirely new programming language which would take a tremendous amount of time.
Total Score: 18

- **CryEngine 3:** The CryEngine has the reverse issue that Unreal Engine has. There is so little foundation and documentation that additional work would be needed to meet basic requirements. Furthermore, no one on the team currently knows how to use LUA.
  Total Score: 17

Figure 1 - Desktop Application Alternatives Comparison

Characteristics Score	**Unity**	**Unreal Engine 4**	**CryEngine 5**
Interactive	**Score: 5** Very basic developer UI allows for everything we need. **Complications:** N/A	**Score: 3** The Blueprint Engine is not easy to learn or interact with, but Unreal does have all necessary abilities. **Complications:** B : A learning curve that would take longer than two hours(-2)	**Score: 3** The developer UI makes things really easy to make, arguably easier than Unity but everything has to be made from scratch to use as templates later. **Complications:** A : Interactive objects need to be made from scratch or are not built-in(-2)
Rendering	**Score: 5** Out of all the options, it has the slowest rendering and most limited capabilities, but it is the easiest to design graphics for. **Complications:** N/A	**Score: 3** The learning curve of using graphics options was made difficult because of the Blueprint Engine. Also the capabilities are way above and beyond what we need. **Complications:** B : A learning curve that would take longer than two hours(-2)	**Score: 3** CryEngine uses Lua which no one on the team has ever used before. Furthermore, creating content for visual display is notoriously difficult in LUA according to the community. **Complications:** A : Cannot display basic geometric shapes without additional work(-2)
Offline Functionality	**Score: 5** Capable of doing all functions listed.	**Score: 3** The Blueprint Engine is not easy to learn or interact with, but Unreal does have all necessary abilities.	**Score: 5** Capable of doing all functions listed.

Characteristics Score	**Unity**	**Unreal Engine 4**	**CryEngine 5**
	Complications: N/A	**Complications:** B : A learning curve that would take longer than two hours(-2)	**Complications:** N/A
Programmer Ease of Use	**Score: 4** Language: C# This is one of the smaller programs, but has a large amount of documentation in proportion to its capabilities. **Complications:** A.i : One team member does not know the language(-1)	**Score: 4** Language: C++ Only option out of the three that has a mandatory engine, but has documentation to compensate. **Complications:** A.i : One team member does not know the language(-1)	**Score: 1** Language: LUA Third party and community support and documentation for the engine is limited. **Complications:** A.ii : Two or three members do not know the language(-2) B : Documentation is limited, determined by the census of the internet(-2)
Cost	**Score: 5** A free license applies as long as less than $100 thousand per year has not been made on the product. **Complications:** N/A	**Score: 5** Royalty fee on all game sales and usage. Additional charges can be made for more content such as 3D models and 2D images. **Complications:** N/A	**Score: 5** 5% royalty fee per sale after the first $5000 is made. **Complications:** N/A
Total Score:	24	18	17

3.1.4 Chosen Approach

For the desktop application, **Team Gaming Ed selected Unity as its chosen solution** due to being simple enough to understand and strong enough to meet project requirements. Both Unreal Engine and CryEngine have too many complications that would create additional work when compared to Unity. Unreal Engine's Blueprint engine creates a large learning curve that the team is not willing to invest time into. On the other hand, CryEngine is so simple that the framework it is built on is not friendly to new programmers due to needing programmer-built systems for functions that are readily available in Unity.

3.1.5 Proving Feasibility

The team has successfully created a basic menu and login and generally understands how to change scenes. The team will create a demo to highlight the drag-and-drop functionality. This will be completed for the requirements specification document by April 15th.

3.2 Website Application

The website application is a key design decision because it will serve as the core piece of software for all course material provided to the students. Databases for all these applications will be manipulated using Microsoft's .NET framework which is discussed in more detail in section 3.4.

3.2.1 Desired Characteristics

An ideal solution would be a technology that helps us complete the project but does not compromise our understanding of the code. The team also needs a technology that is free and easy to install on all operating systems. Each category will start at five points and each reduction will remove a set number of points outlined in each characteristic of this section. With a total of three characteristics, the maximum possible score an alternative could achieve is 15.

- **Ease of Use:** A well thought out and easy to read structure will be maintained on the website application to preserve ease-of-access and stability. Technologies are needed that the whole team can use and operate to create a fantastic looking piece of software.
  Possible Complications:
  A) Costs money (-2)
  B) Not open source (-2)
  C) Difficult to install (-1)

D) Team member does not know the language (-2)

- **Shared Code Libraries:** The team does not want to write everything from scratch. It would be more efficient to use existing libraries. Using shared code can create consistency across teams and projects because the team has a unified way to build projects.
  Possible Complications:
  A) Code libraries are not extensive enough for application (-3)

- **High Security:** High security is important because the team does not want users' credentials to be stolen. It already has built-in Windows authentication, which can be used to make secure and safe applications. Frameworks are great at establishing a secure, reliable connection between host and client while providing excellent cryptography.
  A) Non-consistent security functions across all pages (-2)
  B) Security requires lots of maintenance (-2)

3.2.2 Alternatives

- **.NET frameworks:** A developer platform made up of tools, programming languages, and libraries for building software. First developed by Microsoft in 2002 and primarily runs on Microsoft Windows. This framework is also open source which is important because the team can manipulate the software to change its functionality.

- **React:** React is one of Facebook's first open source projects. React was developed in order to create components for web applications. React is open source and first designed by Jordan Walke who was a software engineer at Facebook at the time. It was first released to the public in 2013 and their latest release came out in March 2021.

- **HTML/CSS:** HTML is a language for describing the structure of Web pages. CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. HTML is easy to use and very popular for beginner programmers.

3.2.3 Analysis

.NET frameworks and React JS are very similar. CSS/HTML definitely does not provide us with enough power in order to efficiently store data. The team definitely needs a framework otherwise it would take too much time to start from scratch. React JS uses JavaScript which is a challenge for our particular group because we are not very familiar with JavaScript. .NET Frameworks uses the programming language C# which we are all familiar with. .NET frameworks also have a ton of code libraries which provide advanced security. .NET frameworks are easy to set up and

also provide enhanced security features which are important to the project because it does not allow credentials to be compromised.

- **.NET Frameworks:** Provides enhanced security and extensive code libraries. Also open source and free to use. Developed and maintained by Microsoft.
  Total Score: 15

- **React JS:** Free and easy to use but requires extensive knowledge of JavaScript which the team is not familiar with. Does provide extensive security features and has code libraries available for use.
  Total Score: 13

- **No Framework:** After researching the team discovered that a framework is required in order to successfully secure and manage our website application. Without a framework the team would need to start from scratch.
  Total Score: 8

Figure 2 - Website Application Comparison

Characteristics Score	**.NET framework**	**React**	**HTML/CSS**
Ease of Use	**Score: 5** Easy installation Used by many companies Maintained by Microsoft Free and Open Source **Complications:** N/A	**Score: 3** Also easy to implement and used by many companies Free and open Source **Complications:** D: One team member does not know the language(-2)	**Score: 5** Very customizable and creates easy to read code, but it is very difficult to maintain **Complications:** N/A
Code Libraries	**Score: 5** Extensive set of class libraries. Provides consistency across all web pages **Complications:** N/A	**Score: 5** Creates functional and impressive-looking applications using the code libraries. **Complications:** N/A	**Score: 2** Has limited code libraries Without a framework you have limited interactivity, functionality, and maintainability **Complications:** A: Code libraries are not extensive enough for the teams website application(-3)
Security	**Score: 5** Provides developers with an easy-to-use toolset to implement powerful authentication, authorization, and cryptographic routines. **Complications:** N/A	**Score: 5** Does not have strong default security setting and becomes vulnerable to security slips, but is more secure than most frameworks **Complications:** N/A	**Score: 1** Does not allow for best practices to manage risks that may arise. **Complications:**

Characteristics Score	**.NET framework**	**React**	**HTML/CSS**
			A: Non-consistent security functions across all pages(-2) B: Security requires lots of maintenance(-2)
Total Score:	15	13	8

3.2.4 Chosen Approach

The team has **chosen to use the .NET framework for storing and manipulating data.** This framework is popular amongst many enterprises in industry today. This framework has numerous libraries that can handle the technical bits so the team can stay focused on more important tasks. .NET framework is free to use, easy to install, and is also open source. React had a great set of libraries to choose from but the team was not familiar with React compared to .NET frameworks. The third option of not using a framework would have been too time consuming without the support of extensive code libraries.

3.2.5 Proving Feasibility

The team will create a demo showing various assignments being submitted and updated in the grade book for the Technical Prototype Demos assignment due at the end of April. The team also plans on having an example demo of a login by a student.

3.3 Database

A database's main functionality is to hold information about some application that's running in an environment on a server. The types of information held in a database consist of strings (which are just sentences or words), integers, doubles, and float values. These values are used to hold information about a user, the data for the website, and other things that vary from application to application. In some cases, this information is held in large tables with rows and columns. Other databases use more sophisticated measures (non-relational database) to hold information not in tables.

3.3.1 Desired Characteristics

The database that we need for this project could mostly be any type of database. It just has to hold information in a way that is accessible to the website/desktop application and the user. A non-relational database, or relational database could be used in our application to store information. However, these databases are not always the same because some cost more to use. Also, the complexity of moving information around in some databases is harder to learn than others. When scoring our choices of databases, it is easier to start by giving a total number of points (5) to each characteristic and then reducing points as problems and inconveniences arise. We will be taking away points from each desired characteristic (listed below).With a total of 4 characteristics, the maximum possible score an alternative could achieve is 20.

- **Fast allocation of memory:** Our database will need to iterate through tables in an efficient manner. Slow databases are inconvenient for the client because of long wait times.
  Possible Complications:
  A) Requires additional payment to increase read/write speeds (-3)

- **Concurrent Use of Database:** In some cases, users will be accessing data at the same time. If the teacher wants to edit the course, they would need to update the database. It is important to make sure that the database is concurrent so other active users will not be affected. In our code, we need to make sure that it is not possible for the teacher to edit a table in the database while a student is uploading information to it.
  Possible Complications:
  A) Database has read/write errors due to two people updating the database (- 5)

- **Flexibility:** In some cases, databases are only available for their one company's services. Companies like AWS and Microsoft have their own databases and some are faster than others. This narrow window lowers the amount of options available to just a handful of companies. There are some databases, however, that can work on any system. For the type of information being uploaded, we need to ensure that every database has a table for each type of information.
  Possible Complications:
  A) This database is only available by the company who provides it (-2)
  B) Ability to create, change, and delete databases (-4)

- **Pricing:** The database we are going to use cannot be too expensive but the more users that our database serves the more it will put pressure on the system. We need to find a good balance between price and performance for our database.
  Possible Complications:
  A) This database is expensive to use (- points depending on price range)

3.3.2 Alternatives

- **CosmosDb:** Azure Cosmos DB is a fully managed NoSQL database service for modern app development. This technology is solely for Microsoft Azure. This database does offer super fast and reliable information, more than a traditional database. This is because each piece of data is broken down into logical partitions and each partition is given a certain amount of R/U's (Request Units that determine speed).

- **Amazon RDS:** Amazon's version of mySQL database is called Amazon RDS. Amazon RDS comes with tools like the AWS Management Console, Amazon RDS API calls, and the AWS Command Line Interface. It also supports a wide variety of different SQL databases and they include: PostgreSQL, MySQL, Maria DB, Oracle, SQL Server, and Amazon Aurora.

- **Microsoft SQL Server:** Microsoft's version of a relational SQL database. This database was first released in 1989 but is still functional to this day. It is constantly updated and is one of the most used databases in the world.

3.3.3 Analysis

All of these databases meet the requirements listed in the characteristics section. Some of these databases require some knowledge to use the main portal that controls the services. This is just the website from the Amazon portal that controls your whole system. Once understood, all of these databases are viable options. A custom server could be set up for Microsoft SQL Server in order for it to work. Amazon RDS and CosmosDb would have to be set up on either Azure or AWS. Points were reduced in accordance with section 3.3.1.

- **CosmosDb:** CosmosDb is one of the newest databases to come into existence. It is arguably faster than SQL because it allows you to purchase more (R/Us) to make your database faster. While it does meet all of our requirements for our database this database is only available for Microsoft Azure and our team would have to be using Azure. This is a very expensive service and is mostly used by large companies with a big wallet, but that is not us.
  Total Score: 12

- **Amazon RDS:** This database fits all of the criteria that we are looking for, mentioned above in 3.3.1. This database uses SQL to hold all of its information which is a big plus since SQL is the most commonly used database in the world. This service was offered to us to use for our Capstone project with NAU paying for it. Amazon's web services are very secure and widely used in the world as well. Since this database is offered to us for free to use, it is a good chance we will be using it.
  Total Score: 17

- **Microsoft SQL Server:** This database fits all of the requirements that we are looking for. This is a very viable alternative as it is available in every operating system. The pricing for this database would depend on the server that we are running our database on.

However, this database is very old and there are newer industry tools and services that we could be using.

Total Score: 19

Figure 3 - Database Alternatives Comparison

Characteristics Score	**CosmosDb**	**Amazon RDS**	**Microsoft SQL Server**
Fast allocation of memory	**Score: 2** Has a latency of less than 10 milliseconds when reading data and less than 15 milliseconds when writing data **Complications:** B) Requires additional payment to increase read/write speeds(-3)	**Score: 5** The efficiency of SQL depends on the CPU processing power, so if you pay for more then you get a faster DB **Complications:** N/A	**Score: 5** The efficiency of this database would depend on the system you run it on **Complications**: N/A
Concurrent Use of Database	**Score: 5** As concurrency depends on the code written and not the database, this is unable to be scored. We need to ensure, in our code, that our database is concurrent **Complications:** N/A	**Score: 5** As concurrency depends on the code written and not the database, this is unable to be scored. We need to ensure, in our code, that our database is concurrent **Complications:** N/A	**Score: 5** As concurrency depends on the code written and not the database, this is unable to be scored. We need to ensure, in our code, that our database is concurrent **Complications**: N/A
Flexibility	**Score: 3** This is only available through Microsoft Azure and the services it provides. This is a very popular service **Complications:**	**Score: 3** This is only available through Amazon AWS and the services it provides. However, Amazon AWS is a very popular service **Complications:**	**Score: 5** Microsoft SQL Server can be put on almost any server with any operating system. It has been around the longest **Complications:** N/A

Characteristics Score	**CosmosDb**	**Amazon RDS**	**Microsoft SQL Server**
	A : This database is only available by the company who provides it (-2)	A : This database is only available by the company who provides it (-2)	
Pricing	Score: 2 CosmosDb is a very expensive database to use because Microsoft Azure is expensive. **Complications:** A : This database is expensive to use (- points depending on price range)	Score: 4 This service is less expensive than Azure. Depending on your usage it can be cheaper **Complications:** A : This database is expensive to use (- points depending on price range)	Score: 4 The pricing on this varies depending on what server you are using this database on. **Complications:** N/A (depends on what system you run it on)
Total Score	12	17	19

3.3.4 Chosen Approach

The choice of **database that we went with was Amazon RDS for mySQL**. The reason that we chose this database was because it was given to us by our university to use for this capstone project. However, it is important to note that all of these databases are viable for this project. The team was given a SQL database to put all of our tables in so we can store information. In order to allow this system to create multiple courses, a desktop application is needed to allow the creation of a brand new database for us and all of its tables each time a new course is made. Once created, the database will then be able to accept information from Unity. SQL should not pose a problem for data storage to use for this project, but there are alternatives to choose from.

3.3.5 Proving Feasibility

The team will be using phpMyAdmin on local machines to run and test the application while it is being built. Using a software called Xampp, the team is able to simulate a server running on local machines. Once activated, it takes over a local port and runs mySQL through phpMyAdmin, which allows simulating updating a database. The team has used this already in their local machines and set up a local database. For Amazon RDS, the only learning curve would be to use the control portal from the website.

3.4 API

The database will need a set of code that tells it where to put information and how to access it. The database will have to have two separate components that add information to it and therefore two separate API's are needed. One API will be needed for the website and another API will be needed for the desktop application. The API puts information into a database by receiving GET and POST requests from the website/desktop application. It then makes a query string, specifying where it should be placed in the database.

3.4.1 Desired Characteristics

When making a website application, we need to ensure that our requests from the student are properly handled. For the desktop application, our API does not need to return any html back to the teacher, so options are widely available. When scoring our choices of API, it is easier to start by giving a total number of points (5) to each characteristic and then reducing points as problems and inconveniences arise. The maximum possible score an alternative could achieve is 15.

- **Requests:** The API needs to be able to take in URL requests from the website or the desktop application and allocate data based on the type of request. This is also the part of the system that also handles any type of error in a request.
  Possible Complications:
  A) Each request from the API needs to be a separate file (-1)
  B) Ability to handle objects coming in (-3)

- **Non-static html:** Some of these API's listed are able to dynamically make html pages by using objects in the html. A static website with hard coded values would not allow the team to create multiple courses.
  Possible Complications:
  A) Not returning a moldable html page (-5)
  B) Not returning objects in html (-3)

- **Ease of use:** Some of these APIs require knowledge of scripting languages that are specific to each type. As a student, we are unfamiliar with some of these languages and they need to be learned. The architecture of each API would also fall into this category because when you have a working system with multiple parts, it makes it easier to have some sort of structure.
  Possible Complications:
  A) Not being able to see every request being handled on one screen (-2)
  B) Amount of effort to learn (-1)

- **Other notable differences:** Some of these API's are drastically different from one another and the functionalities of them vary. Some of these API's are written in different languages making some easier to understand than others.
  Possible Complications:
  A) i) One team member does not know the language (-1)
  ii) Two or three members do not know the language (-2)

3.4.2 Alternatives

- **.NET framework:** This is an open source, cross platform, implementation for running websites, services, desktop apps, and more on Windows, macOS, and Linux. It holds a multitude of libraries that work for building web applications. This language is written in C# in Visual Studio. Versions of this library can only be run on windows so we will be choosing one that fits our needs.

- **PHP:** Hypertext Preprocessor (PHP) is a widely used, open source scripting language. These scripts are held server-side and as a result, returns information to the browser as

plain HTML. PHP can create, open, read, write, delete, and close tables for mySQL on the server. This has no libraries or framework, so making sure that each PHP file executes with a certain request is important.

3.4.3 Analysis

All of the API's listed above meet the requirements for the characteristics we are looking for. Although it may be challenging to use PHP for the API to our website because we would need to use other tools for our html file like React. (See in 3.2 Website Application above). The ability to have our html files as well as our C# files in the same place is a great advantage that .NET gives us and may be a controlling factor in our decision. Some of the desired characteristics are able to be put in multiple fields (requests and non-static html) and there are other notable differences to be scored.

- **.NET(website):** .NET uses a set of controllers that take in different URL requests and then takes that request and returns a View() of that controller function. For an example: if someone were to send a POST request to www.test.com/account with login information they just typed, an AccountController class with the method GetAccount() would recognize that URL with the POST request, do some processing to make sure your username and password were correct, and then return a View() of that page. A View() is simply a .cshtml file that holds objects in html. .NET calls this ability to use objects in an HTML file, Razor Pages. This allows for the ability to dynamically add html lines to a html file. If the information from one course to another course is different. Also, this framework is what is called a "coupled" website application which means that the backend and front end are very interconnected. So interconnected in fact, that the C# code will be in the same folder as the HTML code.
  Total Score: 13

- **PHP (desktop application):** The language PHP is used to communicate between the database and the Unity app. There is really no set framework for this side of the API, because every request made to the server will be the exact PHP file it needs to execute. This differs from .NET because in .NET the url is relative and not the actual location of the file. PHP is unlike any other language and can easily talk to the database. For example: once a submit button is hit in Unity, the C# code will send a GET/POST request to the URL which holds the location of the PHP file (ie. www.test.com/account.php). The PHP code that is located in that file will then allocate that request, put all the variables where it needs to go, and send it to the database in the form of a string. "SELECT username FROM users WHERE username = $username".

Total Score: 11

- **PHP (website):** For the website, using PHP for the API might be a viable option. One of the advantages of doing this is the team would already be writing an API in PHP for the desktop application. The team is learning how to master PHP for the desktop application. Therefore, it could also be possible to learn more about how to use it for the website.
  Total Score: 5

Figure 4 - API Alternatives Comparison

Characteristics Score	**.NET**	**PHP (desktop)**	**PHP (website)**
Flexibility (requests and non-static html)	**Score: 5** Capable of making Razor pages, passing objects to html pages **Complications:** N/A	**Score: 5** Meets all the requirements to successfully update DB **Complications:** N/A	**Score: 0** Very hard to not make hard coded values. Without help from other dev tools. Structure of files being harder to understand when the project gets complex **Complications:** B) Ability to handle objects coming in(-3) B) Not returning a moldable html page(-5)
Ease of Use	**Score: 4** .NET does take some getting used to, and the learning curve is steep **Complications:** B : Amount of effort to learn(-1)	**Score: 3** There are plenty of videos online to help developers write in PHP **Complications:** A : Not being able to see every request being handled on one screen(-2) B : Amount of effort to learn(-1)	**Score: 2** There are plenty of videos online to help developers write in PHP **Complications:** A : Not being able to see every request being handled on one screen(-2) B : Amount of effort to learn(-1)
Other Notable Differences	**Score: 4** ● C# ● Visual Studio gives template to start	**Score: 3** ● Used very often in industry ● PHP scripting language	**Score: 3** ● Used very often in industry ● PHP scripting language

Characteristics Score	**.NET**	**PHP (desktop)**	**PHP (website)**
	Complications: C.i) One team member does not know the language (-1)	**Complications:** C.ii) Two or three members do not know the language (-2)	**Complications:** C.ii) Two or three members do not know the language (-2)
Total Score	13	11	5

3.4.4 Chosen Approach

Our application will be using **.NET and all of its tools in order to communicate with our website**. This is the best solution because it is a very efficient and reliable application. This type of framework is used by companies today and is constantly evolving. Since it is so fast and new, it would be our best favor to learn this for experience after college. **PHP will be our API for communicating with our Unity application.** This is a simple and easy solution because there are many videos online showing how to do this.

3.4.5 Proving Feasibility

The team has already created a basic menu and login screen in Unity which uses PHP files to update the information on our local database (See 3.3 Database, Proving Feasibility). Using PHP is very simple and the requests to make it are simple since it is just the location of the file. Using .NET will prove a challenge to master in order to get this application off the ground. There are lots of videos online to help with developing .NET applications in AWS and we will be referencing those very often. Also, the team already has simple structures and skeleton’s from other team members' previous website’s to look at for inspiration.

4.0 Technology Integration

With all of these pieces put together, this will become a fully functional program that will allow students to take a course built by their teacher. Once completed, this program will be very efficient and will allow for the creation of any course at NAU through teachers building the course in Unity and students taking it through a website. Now that we have talked about every individual aspect of the application, let's take a look at it as a whole.

4.1 Where is the code?

- **Unity-** Unity uses its own library in C# for sending and receiving requests. The C# code to send and receive requests will be on the machine you download the application on. When unity sends a request to the server, it is actually looking for the location of the PHP file on AWS. If we were to update account information from Unity, we would connect to a URL similar to this name, "https://whateverURL.com/Unity/updateAccount.php". Unity then sends information through a POST request to this URL. The PHP file is actually in charge of seeing whether or not the request is valid and/or if it can do anything with it.

- **PHP-** The PHP code is held and stored in the server. The code is in charge of telling whether or not a request is valid through this code: $_POST[] or $_GET[]. The actual location of the file in the server is what our Unity application is looking for.

- **.NET-** The C#, CSS, and CSHTML files will be stored on the server. The C# files are in charge of being hooked to a URL link and a type of request. For example, a method GetAccount() would be linked to a GET request *and* the URL "/account/getAccount". When our website has a clickable link to get account information they will click on it and it will lead them to that link. Which will then hit that GetAccount() method linked to that URL *and* the GET request. Once inside the method, the C# code will access the DB and get any information about it and return a View(). A View() is simply a cshtml file with the *same name* as the method that is calling it. So the html for the account that the user sees would be called GetAccount.cshtml. A View(result) can also be returned with an object in it and this is how you dynamically print items in the html file.

4.2 How it works together

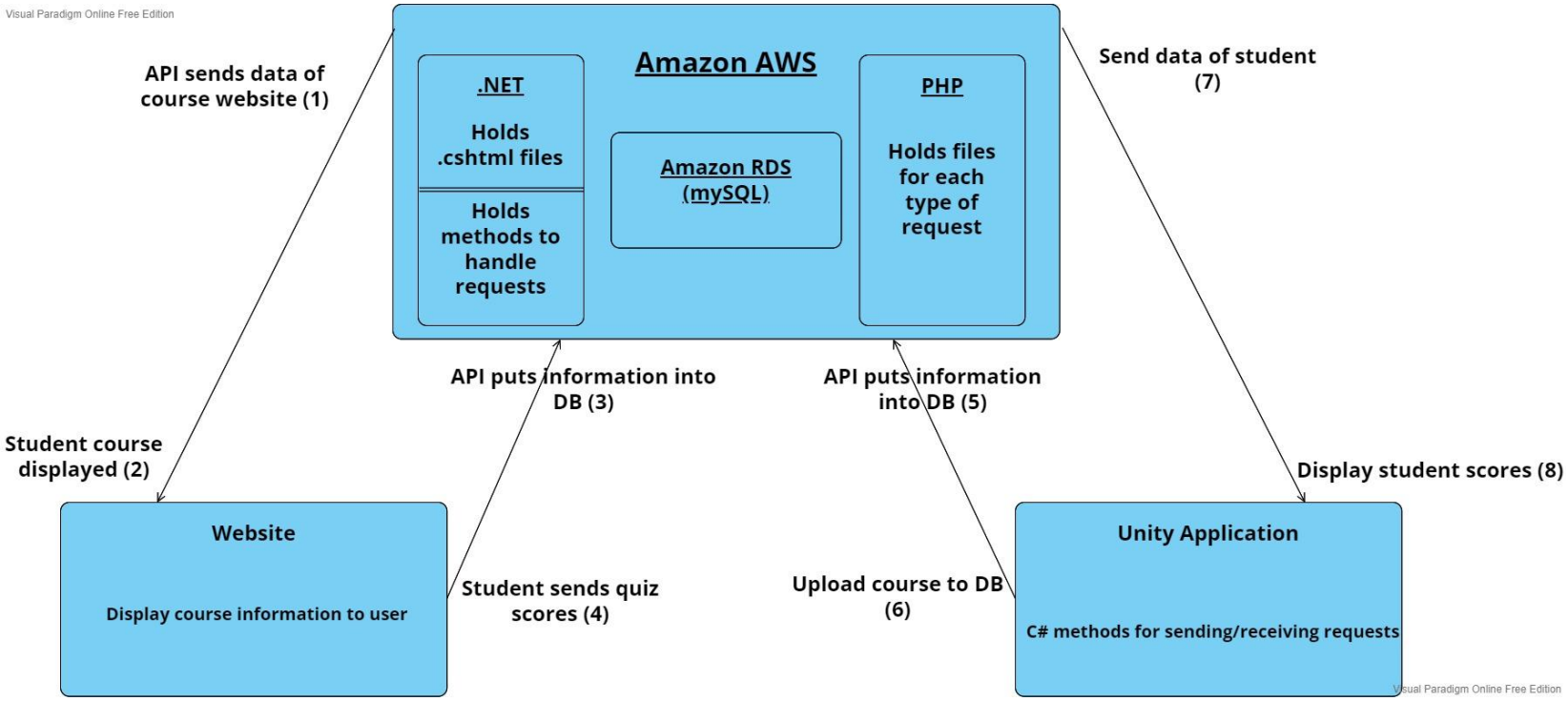
Please reference Figure 5: Diagram of the working application

The teacher will use the Unity Application to first create a course locally. There will be a main line going through the screen to indicate how the flow of the course will work. They can drag and drop icons to determine the next phase for a student to complete. These icons would consist of tests/quizzes and from there they can edit the questions, weight, and total score of the assignment. After the course is completely built on Unity, the teacher will have to connect to wifi and submit the course with a button. (6) We will write code in Unity to allocate which information needs to go to which PHP file to upload to the database. Lets say, for example, that we are updating the quiz count. There needs to be a PHP file that would be available to update said table on the server so Unity can connect to it. Once the request is sent, PHP will put the information into each appropriate table for it to be used by the website. (5)

Once the data for the course is in the database the website can now be used. The student then loads the website through a GET request, (1) which will include information about the website, the website will be loaded with quizzes, tests, the marketplace settings, and how courses are unlocked. (2) Our cshtml file will dynamically add more or less information depending on it existing in the database. If the course were to only have 4 quizzes, then the table displaying them would only have four rows. The student then clicks on the link for the quiz, makes another GET request to the .NET API, and the API responds by sending the quiz data. After completing the quiz, the website then sends a POST request to our server telling it that it needs to update information. (4) The API will have a method linked to some URL for this request and will update the information in the database. (3)

The teacher is also able to see student scores from the application. This is the only GET request that our desktop application will be using.(7) Our application will make a request to our server, find the php file location, and will display the scores of each student to the teacher. (8)

Figure 5: Diagram of the working application



5.0 Conclusion

Many different LMS's are used all over the globe for educational purposes. The main issue with the BbLearn LMS at NAU is that it is not very interactive. Gamifying a course would be a great way to increase student engagement and this requires the LMS to be interactive for the teachers. The problems our client is having with BbLearn is that it requires too much manual labor and doesn't allow for much customization of a course. Without the proper customization tools a course is much more difficult to gamify. The purpose of the desktop application is so that the teachers can easily customize their course. While the students on the other hand will be using the website application to take the course. All data within the database will be shared between both the desktop application and the website application. The API will help manage the interactions between the database, desktop application, and website application.

The framework for the website application will be .NET Frameworks. This is a free and easy to use software which has extensive code libraries to make coding more manageable and consistent. The team will be using Unity engine to develop the desktop application. Unity is a cross-platform game engine developed by Unity Technologies which facilitates the creation of stunning looking games. PHP will be used to save and update information to our database.

Our Tech Demo assignment due at the end of April will be displaying some functionally of our project. Both the desktop application and the website application will both have their own demos.

A new LMS is required to make a gamified course because our client spends way too much time trying to manage his gamified course using BbLearn. Our new application will be easily customizable and be able to be used for a variety of different courses. The team hopes to revolutionize the way future LMS's are developed in the future.

6.0 References

[1] Arora, Simran Kaur. “Unity vs Unreal Engine: Which Game Engine Should You Choose?” *Hackr.io*, 18 Nov. 2021, https://hackr.io/blog/unity-vs-unreal-engine.

[2] Dealessandri, Marie. “What Is the Best Game Engine: Is Cryengine Right for You?” *GamesIndustry.biz*, GamesIndustry.biz, 5 Jan. 2021, https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-cryengine-the-right-game-engine-for-you.

[3] “Introduction to .NET Framework.” *GeeksforGeeks*, 21 Feb. 2022, https://www.geeksforgeeks.org/introduction-to-net-framework.

[4] “Introduction to Azure Cosmos DB.” *Introduction to Azure Cosmos DB | Microsoft Docs*, Microsoft, 30 Mar. 2022, https://docs.microsoft.com/en-us/azure/cosmos-db/introduction.

[5] “The Good and the Bad of .NET Framework Programming.” *AltexSoft*, AltexSoft, 28 Feb. 2020, https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming.

[6] “What Is .NET Framework? A Software Development Framework.” *Microsoft*, https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework.