SciKids

Technological Feasibility Report

11/04/18

SciKids

Sponsor: Elizabeth Glass

Mentor: Austin Sanders

Samantha Earl, Claudia Coronel, Gwen Morris

Table of Contents

1. Introduction

STEM is a program of study that educates students in four areas: science, technology, engineering and mathematic. According to the U. S. Department of Commerce, STEM occupations are growing at 17% annually, while other occupations are growing at 9.8% annually. With a growing need for STEM individuals and the immense careers that use STEM characteristics, there is a desire for more interest in the field. Science, technology, engineering, and mathematics plays an important role in bettering the future for the U.S. This project we are developing heavily involves STEM and attempting to show the benefits that STEM provides. Overall, the main purpose of this project is to heavily encourage individuals to consider a career that involves STEM. With a student development team of Samantha Earl, Claudia Coronel, and Gwen Morris along with our mentor Austin Sanders, we plan on making a gesture-based software with STEM. Our client, Elizabeth Glass, is the director of career development for NAU and was responsible for creating the project proposal. As mentioned earlier, the project is a gesture based engagement station for STEM recruiting from kindergarten up to college age students.

The goal for the team is to create an interactive system where students can choose and feel supported through their educational career in a STEM related field. Since there is a wide age range, there will be different modules for different age sections. Younger groups will focus more on interactive games to pique interest. This can include content like matching games to get kids moving and exposed to STEM related material. Kids are normally exposed to what STEM is around the time they have decided what they are interested in or not. Older aged groups will include resources geared towards career development. This content will include tips for interviews and resumes. In addition, the content will provide information that older individuals will need as they search for employment and exploring the interests they developed as children into a career path to follow.

This document addresses the technological challenges that we are attempting to solve for this project. The document will be an analysis of the feasibility for key elements of the project. First, we will be identifying technological challenges. Here the major design decisions are outlined. The section proceeding is the technology analysis. In this section, the challenges are presented with the functionality that is necessary for the project. Possible solutions to the challenge are then presented. The technology integration is the next section. This shows how our chosen solutions for each technological challenge come together to create a coherent system that meets the needs of the project. Lastly, we will provide a conclusion to end our feasibility report.

2. Technological Challenges

Based on the needs of the project, there are several technological challenges that we have identified. The team has decided that we will need to have a portable device that is easy for the everyday person to set up. We will need a way for information to be saved and displayed on both a gesture-based system and an average PC. In addition we will need hardware to be able to detect visual images.The hardware will need to be able to recognize gestures and the software would need to be able to interpret them appropriately. Because of this, the hardware and software components are tied together. The features of the combination of hardware and software include the ability to provide gaming features, a sensor bar to pick up the gestures, and provide the ability to create a navigational framework in the modules as a place for games to live.

For this project, we will need a feature that is a highly graphical user interface. For this interface to work, we will need it to be able to accept gestures as event triggers. This GUI needs to be able to work across different platforms, one being a PC and the other being a gesture-based system. The GUI system must be compatible with the hardware and software. It is responsible for providing the feel of the system. Modules for the different age groups will have different designs to target certain age groups. In addition, the interface must also support the gestures that people will use to interact with for the navigation and gaming aspects.

A feature that our sponsor would like is the ability for individuals to save data relating to their gameplay. This information can be stored and accessed through a login system. The login system allows for users to create a account for the module. The system will allow people to create an account with a username and password. The correct combination of password and username will be used to access the information relating to the user. This system needs to have some form of validation. The validation is used to help force people to enter information that meets certain requirements. For example, the email section should take in text that resembles an email addresses not the text that represents a username or date. The database can store information like how many points a user has acquired in game play or what the account users personal high score is for individual games.

3. Technology Analysis

This section will include an in-depth analysis of the technological challenges we have proposed in the previous section. There will be several alternatives for each challenge presented in this section. Each alternative will have an in depth description of the characteristics of these solutions. Additionally, we will select the best option for our team to use based on selective criteria.

3.1 Full-Body Detection

In this section, there will be two important technologies that will be analyzed - the motion-sensing camera and its corresponding SDK. These technologies should address certain features that will be part of our final solution. One of the most critical features, as stated by the client, is portability. The camera should be easy to transport, and the software should be simple enough to set up on a variety of computers. We will be comparing the physical size of each cameras and some the SDK's PC requirements (such as the operating system requirements). Several specifications of each camera will be compared and contrasted (such as the camera's depth range and field of view). The practicality of obtaining each device will also be put into consideration, as we want our client to purchase multiple units with confidence if she so wishes. We will be comparing the languages each SDK supports and what development platforms are available. This will determine the learning curve for us and future programmers. Documentation of the SDK and its supported development platforms will also play an important role, as it will help us learn how to use the software faster. The SDK will also partially dictate what development platform to use for backend processes. Because our client wants content to be displayed in an interactive and engaging manner, a lot of the content will have to be presented as games. To do this, we will need to look at how the development platform supports game creation.

When the project was first introduced to us, the main example used of a gesture-based device was the Microsoft Kinect. However, gesture-based systems do not necessarily mean a kinect-like camera (for example, a smartphone is considered to be gesture based). After talking to the client, we decided that it would be best if the system supported fully-body detection. Therefore, any devices that require a remote or something similar would not be considered. We have narrowed our choices down to three devices (along with their corresponding SDKs): the Microsoft Kinect v2, the Intel RealSense D435, and the Orbbec Astra Pro. Our choice of the device and SDK will in turn narrow our development platform choices, which will also be discussed.

3.1a Microsoft Kinect & Kinect for Windows SDK 2.0

The following refers to the Kinect 2.0. The Kinect's camera takes in color image frames from its RGB camera and depth image frames from its depth camera, which developers can use to interact with programs with the help of the SDK. When using the device, the user must be 0.5 meters to 4.5 meters away from the camera, with the "sweet spot" being between 0.8 meters and 3.5 meters. The device can track up to six people as whole skeletons. The kinect's dimensions are 9.8" x 2.6" x 2.63" (258mm x 66mm x 66.8mm). The Kinect's angle of vision is 70° horizontal x 60° vertical. While not a requirement from the client, the Kinect's abilities to track up to six people is one of it's strongest advantages when having a classroom setting in mind. (More information can be found on https://developer.microsoft.com/en-us/windows/kinect).

The Kinect for Windows SDK is compatible with other Windows developer tools, primarily Visual Studio. However, version 2.0 of the SDK offers developers more cross-platform support. Unity Pro packages are available to make Kinect-based Unity apps and NuGet packages are available for the .NET framework. This SDK also offers native APIs for writing C++ code and offers a visual gesture builder. The visual gesture builder uses machine-learning for gesture detection, allowing developers to define specific gestures. The system requirements include having an operating system of Windows 8 or higher, USB 2.0, and at least 4GB of RAM.

A very important downside to the Kinect is that the Kinect v2 for Windows and the Kinect SDK were discontinued in 2015, and the equivalent XBox One Kinect was discontinued in 2017. On top of this, the XBox One adapter for Windows was discontinued in January 2018. Consequently, obtaining the device would be made more difficult and the client could not upgrade devices in the future. On Microsoft's Kinect developer website, Microsoft recommends the Intel RealSense depth cameras as an alternative, as Microsoft is working with Intel to transition out of the Kinect.

3.1b Intel RealSense Depth Camera D435 & Intel RealSense SDK 2.0

Intel's RealSense is a strong contender to the Kinect. Like the Kinect, Intel's RealSense makes use of depth and RGB sensors along with an infrared projector. The camera's range is approximately 0.11 to 10 meters, however accuracy can vary based on calibration and lighting conditions. The camera is significantly smaller than the Kinect, with dimensions of 99mm x 20 mm x 23 mm. One of the client's main requirements is portability, so this camera's size works to it's advantage. The connecter is a USB 3.0 Type-C, so some computers may require an adapter for the camera. The depth stream frame rate is up to 90 frames-per-second and the depth field of view is 85.2° horizontal x 58° vertical (More information can be found on https://realsense.intel.com/stereo/).

The Intel RealSense SDK 2.0 is readily available on GitHub, which includes the Intel RealSense viewer, a depth quality tool, debugging tools, code samples, and wrappers. The languages that the SDK supports are Python, C, C++, and C#, while wrappers that are supported include Unity, ROS, and Matlab. This is able to track 19 points on a full body skeleton. The SDK is well documented and provides sample code for incoming developers. This will aid us in learning how to create programs for the RealSense and will also allow us to quickly create prototypes. (This can all be found on https://github.com/IntelRealSense/librealsense). The system requirements are to have either Ubuntu 16.04 or Windows 10.

The Intel RealSense is still being manufactured, so units can be purchased directly from Intel. The unit also has some credibility to it by being manufactured by a widely known company. Intel is also working with Microsoft for Kinect developers to transition to Intel RealSense. Some cons would be the inevitable learning curve that comes with the wrappers and languages the SDK supports. While everyone in the group has some experience in Python and C, we all have limited C++ and C# experience. We would also have to learn a wrapper that we have never used before. We are planning to add gamification to some of the content included in our product, so a preferable wrapper offered by the RealSense SDK would be Unity. This poses a challenge, as none of the team have experience with Unity. However, Unity has created their own developer page for the Intel RealSense (found https://unity3d.com/partners/intel/realsense), so this resource would greatly help us in developing our product.

3.1c Orbbec Astra Pro & Astra SDK 2.0

When looking for other alternatives for the Kinect and Intel RealSense, an article by Vangos Pterneas (CEO to LightBuzz Inc), recommended the company Orbbec. Pterneas says “As a business owner and Software Engineer, I have to make a choice that covers the business needs of my clients and customers...Orbbec is, by far, the most reliable option right now.” (https://pterneas.com/2017/10/25/kinect-dead/). The Astra Pro’s camera depth range is from 0.6 meters to 8 meters while its field of view is 60° horizontal x 49.5° vertical. The size of a unit is 160mm x 30mm x 40mm. This device requires a USB 2.0 port.

The Orbbec Astra Pro runs with the Orbbec Astra SDK. This tracks 19 joints on a full body The SDK offers native C and C++ API along with .NET/C# and Java wrappers. A Unity wrapper and Visual Studio support is also available for the SDK. The SDK is available for Windows 8 or later with 4 GB RAM, Ubuntu 14.04 or later with 1 GB RAM, macOS 10.6 or later with 4GB of RAM, and Android OS 4.4.2 or later with 512 MB RAM. The Orbbec Body Tracking Library is free to use until January 31, 2019, and this deadline could pose a potential problem with Orbbec.

Like the Intel RealSense, Orbbec has a similar con in its learning curve. We will still have to use an unfamiliar development platform and languages. The main difference is that there seems to be better documentation for Intel’s SDK than Orbbec’s.

3.1d Chosen Approach

To quickly summarize the features of each of the cameras and SDKs,

Device	**Size**	**Depth Range (in meters)**	**Field of View (horizontal x vertical)**	**Device Acquisition**
Microsoft Kinect v2	258mm x 66mm x 66.8mm	0.5 to 4.5	70° x 60°	Must buy from third party vendors or buy the XBox One equivalent.
Intel RealSense D435	99mm x 20 mm x 23 mm	0.11 to 10	85.2° x 58°	Can be bought directly from manufacturer.
Orbbec Astra Pro	160mm x 30mm x 40mm	0.6 to 8	60° x 49.5°	Can be bought directly from manufacturer

SDK	**Languages used/supported**	**Development platforms**	**Learning curve for our team**
Kinect for Windows SDK 2.0	C++	Visual Studio, Unity Pro	High - no experience with C++ or listed development platforms.
Intel RealSense SDK 2.0	Python, C, C++, C#	Unity, MATLAB, LabVIEW	Medium - some experience with C and Python, but not with the listed development platforms.
Orbbec Astra SDK 2.0	C and C++, with .NET and Java/C# support	Unity, Visual Studio	High - some experience with C, but no experience with listed development platforms.

Microsoft's discontinuation of the Kinect has highly affected our decision when choosing a camera. Simply purchasing a device would create some uncertainty in product acquisition, as we can no longer buy Kinect for Windows v2 directly from Microsoft. The closest we would be able to come is buying the Kinect for the XBox One along with a PC adapter (which has also been discontinued, so a third party adapter would have to be purchased). This left our choice mainly between Intel and Orbbec. While they both had similar pros and cons, the Intel RealSense SDK is what ultimately swayed our decision. The SDK is free and has thorough documentation on GitHub, including sample code for learning and prototyping purposes. These resources will aid us in creating a base with a navigation system for games/content, and it will also help future programmers to add to it. As such, we have decided to use the Intel RealSense D435 along with the Intel RealSense SDK 2.0. This choice has narrowed our choices for development platforms down to Unity, MATLAB, and LabVIEW. Unity is a well known game development engine with a large community, giving us plenty of resources to work from. Because our client wants some of the content to be presented as games, Unity is our best option for backend processes (and GUI development, as will be discussed in section 3.2).

3.1e Proof of Feasibility

We will first have to purchase the chosen device, download the corresponding SDK, and download Unity. Afterwards we will have to ensure that the SDK and camera are communicating and that the SDK can be linked with Unity with no issues. Once this is done, we can begin to build our product. We will start by looking into sample code that the Intel SDK provides us. The sample code should give us a quick way of creating a device prototype along with teaching us how to manipulate data taken in by the camera and interpreted by the SDK. This device prototype will consist of a simple button that, when selected by a push gesture, will display "Hello World". This will also work in part with the next section analyzing options about GUI creation. If we are able to create this simple program, it will show that the camera and SDK works properly and Unity is a viable option for backend operations. The team plans on buying just one camera initially, so the ease of transporting the camera between members and setting up gesture-based programs in our own homes will convey the camera's portability.

3.2 Graphical User Interface

When choosing a software for our graphical user interface we need to consider several items. We need to considered the adaptability, navigational abilities with respect to virtual reality, an intuitive system, and community support.

This system needs to be able to provide a web application interface for desktop interaction as well as a gesture based interaction. The gesture based interaction would then be projected on a large wall or screen.

The GUI also needs to be good at supporting 3D features. Some of these 3D features include: swiping, dragging, dropping, levels, etc. These features are also included in external libraries of frameworks.

Lastly the software needs to be something with a manageable learning curve as well as have community support. For a manageable learning curve, the team's main focus should be on the development of the end-product, not on learning the software for the GUI. In community support, the team is looking for a well documented system where questions or similar challenges can be resolved.

3.2a Kivy

The Kivy framework is very powerful for handling everything from widgets to animation, and includes its own language for describing user interface and interactions. It is an open source Python library for the rapid development of applications that make use of innovative user interfaces such as multi-touch apps.

One of the great things about Kivy is that it is a cross platform framework. Kivy runs on Linux, Windows, OS X, Android, iOS and Raspberry Pi. It is also Open Source meaning it is free to use under an MIT license.

Since Kivy's launch is recent (May 2017), this is considered a downside because there is work to be done. Because this framework is fairly new, online help and tutorials are limited.

3.2b QT

QT is a C++ cross-platform application framework and widget toolkit for creating graphical user interfaces. The C++ framework comes with cross-platform libraries and APIs. The website says that QT is simple to use because of its effective drag and drop tools for UI. QT also supports 3D gaming which suits our project's goals.

One of the great things about QT is that it has extensive documentation with a huge user base. Because of this, resources for online help and tutorials are easily accessible.

One of the downsides of the QT framework is the cost. Subscriptions for the QT framework and development tools as well as all the QT APIs start at $459/month. While this is a costly option, the other option another option is using the open source release. While there are some differences in the commercial and open source version, the open source release could still manage for our project goals.

3.2c Unity - UI

Unity is a cross-platform game engine that was developed by Unity Technologies. It was first announced in 2005 at Apple Inc.'s Worldwide Developers Conference as an OSX-exclusive. Unity is an all-in-one editor that has a vast range of graphic designing tools with it's built-in UI system. Unity also supports 3D development which fits our specific needs.

One of the great things about Unity is that it has extensive documentation and help online. Since the community is large, developers have support and may find help quickly online.

One of the downsides of Unity is that its learning curve might be excessive for us to manage within our given timeframe. Unity's primary language is C++, although our team does know a small amount of C++, we are not completely confident in it.

3.2d Chosen Approach

Scale(1-5)	**Adaptability**	**Navigation Abilities**	**Easy to Work With**	**Community Support**
Kivy	5	2	5	1
Qt	5	5	3	3
Unity	5	5	3	5

Table 1

In this portion we will be evaluating each software against our specific criteria as shown in Table 1.

Kivy

- Adaptability - as proven to be adaptable since it is considered a cross-platform framework.
- Navigation Abilities - Since the framework is still in its first version, the navigation abilities exist, they are just not as extensive as Unity or Qt. During research there was little to no evidence of Kivy being used in VR which is primal to our project.
- Easy to Work With - Kivy is primarily written in Python, of which all of the team members are very confident in.
- Community Support - While Kivy is a good framework for building user interfaces, it’s stable release was dated July 2018. Since the release is so early, the community for Kivy does not have much support.

Qt

- Adaptability - Qt has proven to be adaptable since it is considered a cross-platform framework.
- Navigation Abilities - The navigation abilities of Qt are extremely powerful as it is a well developed software. With Qt 3D, VR interfaces are very much possible
- Easy to Work With - Qt is written in C++. While the majority of the team knows this language we are not completely confident in it.
- Community support- While Qt is great choice for developing this product, we would only be using the open source version. Using the open source version and not buying the commercial release means that there is a limited amount of features available to us. There is also certain documentation that we will not be provided.

Unity

- Adaptability - Unity has proven to be adaptable since it is considered a cross-platform framework we are able to publish our end-product in over 25 platforms.
- Navigation Abilities - The navigation abilities of Unity are extremely powerful as well. Unity is a well developed software that has been worked on since 2009. It is also very compatible with a majority of VR hardware.
- Easy to Work With - Unity is written in C++ and C#. Again, while the majority of the team knows C we are not completely confident in it.
- Community support- Unity has a large community support, they have various documentation and tutorials online.

By carefully evaluating all the important factors in developing this project, we decided to use Unity for our graphical user interface.

3.2e Proof of Feasibility

Our goal for proving the feasibility of using Unity is to develop a graphical user interface that interacts with the back-end content of the product. To demonstrate feasibility for this project, we plan on creating a very simple front-end GUI that communicates to the back-end content of our project. Included in the demonstration will be very simple features that will be in our overall project like dragging and dropping items as well as selecting in a 3D way. With the help of the Unity community, documentation, and existing online tutorials we will be well prepared for overcoming this technological challenge.

3.3 Login System

For our project we need to create a login system for our users. The goal of the login system is to allow users to keep track of how many points they acquire while they are playing. The users can also create an account and fill out a profile for themselves to tie information to a name. These needs are satisfied by a basic login system allowing for several possible options to solve this technological challenge. The language we use to setup this database should also be fairly easy to learn and provide a simplistic user interface. The login itself is expected to perform basic validation to ensure that the information that users enter meets certain requirements. This includes providing a valid email and ensuring passwords have certain characteristics to increase its strength

3.3a PHP MySQL

In order for this solution to work, the project would have to implement both PHP and MySQL. PHP is an open source web application development scripting language. This would just create the login interface on the website. The data itself would have to be stored in the MySQL database system. Both languages are open source projects allowing users to access their source code and are free of cost.

There is a wide range of benefits for using PHP and MySQL. Both of these programs are cross platform systems and can work on Linux, Windows, and Unix. MySQL and PHP are both very popular database scripting languages and provides a large support system. These two technologies are capable of producing a user friendly interface and intuitive login system. PHP has useful built in functionality to ensure that account information is acceptable. It can flag incorrectly entered information to help prevent human errors like typos that violate requirements from being saved.

The languages are known for their simplicity, reliability, and ease of use. The popularity and ease of use have lead to several informative free tutorials online for both PHP and MySQL. Some of these tutorials include a walkthrough of how to set up a simple login system like the one the project would need to implement. MySQL is also considered a nonprocedural language allowing for the team to focus on what needs to be done with MySQL not how it needs to be done.

3.3b Unity MySQL

Another option to create a login system is through unity with MySQL. Unity is a cross platform game engine. The language used by this development system is C#. As with the last alternative the MySQL will store the login information in a database itself and will work in tandem with the Unity engine.

Unity is a good choice because of its compatibility with many platforms that are currently available. Some of the platforms that are available include the XBox One and several virtual reality systems. These systems all allow for motion based interaction between the user and the system. Another platform supported by Unity iis the web and desktop systems. This would allow programming to be done in Unity but made accessible for both the mobile hardware system and the desktop version. Validation in Unity would require more upfront programming from the team. We would have to code how to ensure information being entered meets certain requirements. Another option would be to implement a PHP file inside the Unity code that can aid in validation.

The Unity editor itself is supported on both Windows and macOS systems making it compatible with the computers that are available to the team. The Unity editor is available under three licenses. One is free allowing the team to access the editor without needing to access the budget for the project.

3.3c Python

Another option would be to use the Python language to create a login system. Python is a high level programming language that is used for general purpose programming. It is another free and easy to use and install language.

The main benefit of using Python is because the entire team has experience using Python. Python is the language used in Northern Arizona University's introductory computer science course. All of the project's developers are computer science majors and all of us have taken the intro class CS126. Using a language that we already know will reduce the learning curve that the project is presenting.

Since Python is a common language, there are many Python tutorials online for the development team to use for resources. Using a language we are all familiar with means we can focus more on creating the GUI and underlying framework that will support the gamification content. Python has several libraries that can be used to help ensure information that is provided to create an account is correct. These features are not as built in as PHP but we can compensate for this by manually creating the checks using the functions from libraries that are provided with Python.

3.3d Chosen Approach

Table Presenting Alternatives

Login System	**Existing Validation**	**Familiarity (scale of 1-5)**
PHP	Has built in methods for validation	4
Unity	Import PHP Manually check requirements are met for validation	3
Python	Has libraries to ease comparison, but would need to create functions for validation	5

For the creation of the login system, the top candidates are the use of PHP and Python with the backing of an MySQL database to store user information. To code the login system in Unity would add to the learning curve since the group has no prior experience in C++ and C#. Python is a language the entire group has at least one semester of experience in. Python itself doesn’t have as much built in functionality compared to PHP. This would require us to create more helper code to achieve the same results that PHP can provide. PHP provides the most features to allow for ease of validation. It is also common to have PHP and MySQL working collaboratively, creating several resources to help with debugging. PHP provides many benefits that will ease the creation of a login system.

3.3e Proof of Feasibility

The feasibility is the creation of a simple login on Unity. This ensures that MyMySQL will hold data information and tie it to a login attempt in PHP. We create one or two users to test the system. This will validate that the information is stored correctly and that the information can be accessed from the database then eventually displayed into the GUI for the user to interact with.

4. Technology Integration

Currently our complete set-up of the end product is to have the Intel RealSense D435 and the Intel RealSense SDK 2.0 as the low-level hardware and software to help us read user gestures. We will then use Unity, which has an Intel SDK plugin available to developers, to use the gestures taken in by the camera as a medium between the user and the program. Unity will be our main tool for developing backend processes, which will be programmed in C++. We will first focus on creating a navigational tool that will allow users to select games/modules. The navigational tool will have a set of predefined gestures (such as a swiping motion to go to the next page and a pushing motion to select a module). However, once a module is selected, the content developers can choose to define their own gestures based off of the module content. After a navigational tool is created, we will integrate MySQL through PHP to store login information and allow users to keep track of progress in a module. Because Unity offers a MySQL plugin, we should have no major problems in incorporating everything together. As our navigational tools and database are being developed, we will be using Unity to create an easy to use GUI. Regarding future content to be added, it is again up to future programmers to decide their own GUI based off of the content they plan to add.

5. Conclusion

Our minimal viable product must be the base of a gesture-based system designed to spark interest in STEM careers. The system must make use of physical gestures (such as a single-handed push, a swipe, etc). We must have a navigational tool that allows the user to select between different modules. This navigational tool should also be usable for future programmers to add content to. Finally, we must have some kind of database in place for users to store login information so that they can track their progress and points in certain modules.This comes with three main technology challenges - which hardware/SDK to use, what to use to define backend processes and a corresponding GUI, and what database management system to use for login information and progress in modules. Important aspects that the hardware and SDK should have are portability, thorough documentation, and ease of purchasing the hardware. The GUI should be compatible with both desktop and gesture-based interactions, support 3D features, and have a manageable learning curve. The database does not need to be overly complicated. The language(s) used by the database should be simple enough for us to perform basic user validation and allow for the storing of points acquired in the module.

The Intel RealSense camera is physically small and is still being actively produced and easily attainable. The SDK includes in-depth documentation on their GitHub along with sample code for us to use for prototyping. Because of these features, we decided to use the Intel RealSense along with its corresponding SDK. Choosing the SDK consequently narrowed our options for development platforms. We are expected to add gamification elements to the product, which led us towards using Unity for both backend processes and GUI development. The Unity Game Engine is a well known and well documented development tool. It provides several benefits for the project including cross platform capabilities, a visual editor, can render 2D and 3D images, and offers a personal version for free. We also decided that the database does not need to be overly complicated as it mainly needs to store login information and keep track of user progress, so we decided to use PHP and MySQL to create the login system and store information in the database. The built in validation features from PHP provides more usability for the developers.

The elements of PHP, MySQL, Unity, and Intel RealSense are all capable of implementing the requirements for the capstone project. All of the technologies are compatible due to Unity being cross platform and capable of providing a GUI for the project. PHP and MySQL will act in the background for the login system and data storage. These solutions provide some built in functionality to create the functionality for the system.