Project Report

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

May 1, 2024

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno,

Jacob Michaud, Maha Fazli

Virtual Video Modeling on the Social Skills of Adults with Autism

Project Report

Table of contents

1. Introduction

This was a two-semester (Fall 2023 - Spring 2024) computer science capstone project to complete the requirements of a capstone experience at the University of Maine. This was a project for Dr. Sarah Howorth on virtual video modeling of the social skills of adults with autism. Dr. Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS® curriculum was written by Dr. Elizabeth Laugeson.

The following description comes from the website for the UCLA PEERS® Clinic,

> "The Program for the Education and Enrichment of Relational Skills (PEERS®) is world-renowned for providing evidence-based social skills treatment to preschoolers, adolescents, and young adults with autism spectrum disorder (ASD), attention deficit/hyperactivity disorder (ADHD), anxiety, depression, and other socio-emotional problems." (PEERS®, 2023)

The curriculum has been converted into a PEERS® mobile application on iOS. The mobile app contains information, video role-play examples, and practice questions to help users learn social skills. Dr. Howorth was interested in creating the next evolution of the PEERS® app. She believed that users can learn social skills (eg; communication, humor, dating etiquette, etc) more effectively through a VR interface as it provides a more private, immersive and engaging experience. The aim of this capstone project was to create a proof-of-concept VR experience that can be used to solicit funding for further research.

1.1 Purpose of This Document

This document shall encapsulate all of our design and development throughout this project. This document is intended for anyone interested in learning more about the project, as well as potential future capstone students taking on this project. This document will provide an introduction to the project, the client and the team, as well as discussing the completed requirements of this project. The sections in this document will discuss the following: Section 1 introduces the project and the client. Section 2 introduces the team and deliverables created throughout the project. Section 3 focuses on the requirements completed. Section 4 is the project's design process. Section 5 is development of what we used to create the project. Section 6 is what results were achieved at the end of the year, and Section 7 is the future of PEERS® VR.

Included in the document are also Appendices B, C, D, E, F, and G which correspond to the Software Requirements Specification (SRS), System Design Document (SDD), User Interface Design Document (UIDD), Code Inspection Review (CIR), Administrator Manual (AM), User Guide (UG) created throughout the planning and developing of this application. These appendices include the full documents and all information and diagrams contained within them.

1.2 Problem Statement

Building social skills for adults with autism by reading a textbook or using a mobile application is typically not immersive, which can affect the effectiveness. A virtual reality environment would allow this material to be learned in a more immersiveness environment and hopefully result in increased effectiveness of learning these social skills.

2. Project Management

This section will provide an overview of our team's project management strategy, and will cover topics such as: roles and responsibilities, and team management strategies. These topics will further expand on the responsibilities of the team, as well as the successful strategies that have been implemented throughout the project.

2.1 Roles & Responsibilities

Each member of the team has volunteered for and decided on a specific role in order to delegate responsibilities to everyone on the team.

Tristan Cilley	- Team Leader, scrum master, documentation, development
Maha Fazli	- Team Scheduling, documentation, and UI designer
Jacob Michaud	- Lead developer, code review leader ,documentation
Allison Lupien	- Documentation, development, development support
Nick Sarno	- Client liaison, presentations, and documentation

2.2 Team Management Strategies

Our team has held meetings every Monday and Saturday, where we discussed future work, current assigned as well as accomplish necessary work congruently. This has been effective and has made sure that our lines of communication have been strong throughout the creation of this project.

2.2.1. Deliverables

This section will contain information about the deliverables we have completed during the two semesters of Fall 2023, and Spring 2024 semester.

The SRS was created first to present an overview of the entire project, the SDD was then created to describe the system’s design and architecture, system flows, filesystems, and databases. Lastly, using the flow diagrams and layouts stated in the SDD, the UIDD was created as a guide for the system’s user interface.

The following semester our deliverables included the CIR, the AM, and the UG. Each document was created in succession, using information derived from development. The Code Inspection Report is a record of all aspects of JamTech's code review process. The Administrator Manual is to inform administrators on how to run and maintain the application in order to use this application as a research tool. The User Guide includes explanations of all relevant features, operations, errors, and instructions pertaining to PEERS® VR.

Deliverables for Fall 2023 - [COS397]

An electronic file containing the following:

** Documents are available on Github **

- Systems Requirement Specification, Version 2.0 **(made available: Nov 1, 2023)**
- System Design Document, Version 1.0 **(made available: Nov 15, 2023)**
- User Interface Design Document, Version 1.0 **(made available: Dec 5, 2023)**
- Critical Design Review Document, Version 1.0 **(made available: Dec 15, 2023)**

Deliverables for Spring 2024 - [COS497]

An electronic file containing the following:

** Documents hosted on Github will be updated regularly **

- Critical Design Review Document **(made available: May 3, 2024)**
- Code Inspection Report **(made available: May 3, 2024)**
- Final Project Report **(made available: May 3, 2024)**
- User Guide **(made available: May 3, 2024)**
- Administrator Manual **(made available: May 3, 2024)**
- All source code **(made available: May 3, 2024)**
- The executable program **(made available: May 3, 2024)**
- Any other software required for installation and execution of the delivered program. **(made available: May 3, 2024)**

3. Requirements

This project was developed during the Spring Semester (January-May 2024) . We were able to complete the following features: Video View, Lesson View, Quiz View, Content Database, Curriculum View and Tutorial. For this project to be handed off to the client all of these tasks needed to be completed by the end of the spring semester. Each task was assigned in a

consecutive 2 week sprint where tasks were given at the beginning of the sprint and re-evaluated at the 1 week mark.

There are two general sides to development in Unity. The first is making the objects and environment look clean, and the second is ensuring the scripts function correctly. We knew we had completed a design requirement by comparing our Unity project to the user interface mockups as seen in the UIDD. For each new feature, the primary developer waited for approval from the majority of the group before pushing the feature to the GitHub repository. It is more complicated to determine if our scripts function correctly enough to be considered done. This process involves manual testing, unit testing and system testing to make sure functionality works as defined in the SRS and SDD documents. Over the course of development, some features changed from their original design in the SDD document. These changes generally meant a variation in the method of implementation, but, as long as the feature still met the requirements of the SRS, it was approved and marked as complete. These changes in development strategy were largely communicated informally during team meetings and documented in the Discord server.

Our requirements were defined in the SRS document we published in the Fall of 2023. We were generally successful in implementing our requirements, but there were a few requirements we could not implement, including text to speech and user login. With text to speech we discovered there were no reliable options that were guaranteed to work. Additionally, many options required a monthly or annual subscription fee. After consulting our client, we decided to abandon the feature. For the user login, we did not have enough time to implement the login system. This was largely because it turned out to be much more complicated than we had planned for. We came to the decision that it was unnecessary for our target proof of concept. The amount of development time it would have required was better spent polishing and debugging more essential features. We were successfully able to implement the tutorial, curriculum view, dynamic lesson view, dynamic video view, and a dynamic quiz view with smooth transitions between each. In addition, there is functionality for displaying 360° videos that can be used once the video content has been recorded with 360° cameras. For now, the video content functions with the same 2D videos that were used in the mobile app.

4. Design

This section gives an overview of our design process including documents and diagrams created as well as an introduction to our application's user interface design with specifics from the final application.

4.1. Design Process

In order to design this project we planned out and created multiple design documents in order to fit the clients requirements. We started by creating the SRS document, including use case diagrams to determine what features needed to be implemented for the client, and how those features would be used. In our SDD, we created a UML diagram that served to represent all the

classes, methods, variables, and objects that we would need to create for the project, and how they would interact with each other. The UIDD helped establish what we wanted the application to look like including mockups of the various views that we wanted to include. In the spring semester, we started implementing these ideas and we continued to follow the overall structure and format created in these documents. However, we significantly changed our code to adapt to meet the requirements as the UML diagram did not translate well into development within Unity. We designed and developed our project to have the potential for future scalability and room to implement new features, but most importantly as a proof of concept that will allow our client to be able to use it for research.

4.2. User Interface Design

This section of the document serves as a comprehensive introduction to the project's user interface (UI) design. It begins with a general overview of the various UI elements incorporated in the project. Each UI screenshot is individually presented with a summary of what each screen in the project does.

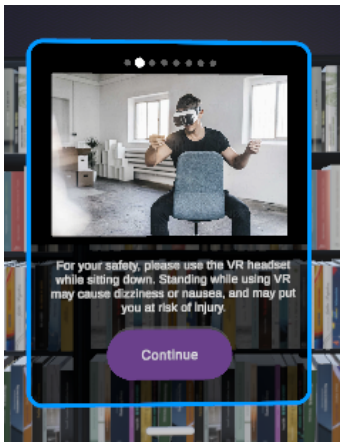


Figure 1 Tutorial: *This is where the user can view a tutorial on how to use PEERS® VR.*

Summary: Once PEERS® VR is launched, the user is presented with a tutorial on how to use VR , if they are unfamiliar with the technology.

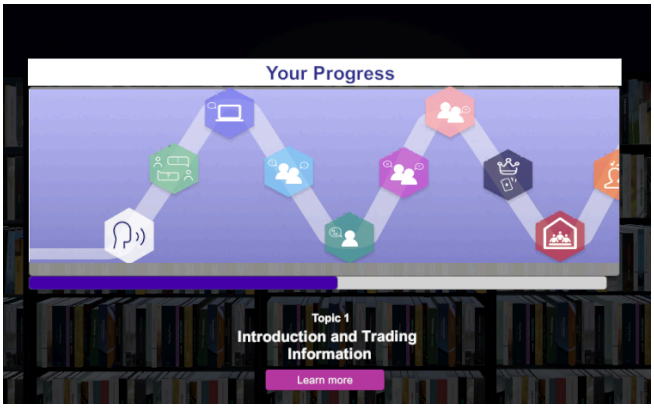


Figure 2 Curriculum View : *This is where the user can view all 18 lessons provided by the curriculum.*

Summary:

From the curriculum map, the user can click on any lesson to be brought to the lesson clicked in. The user can choose which lesson they would like to interact with by clicking it, and then by clicking on the 'Learn More' button at the bottom of the screen.

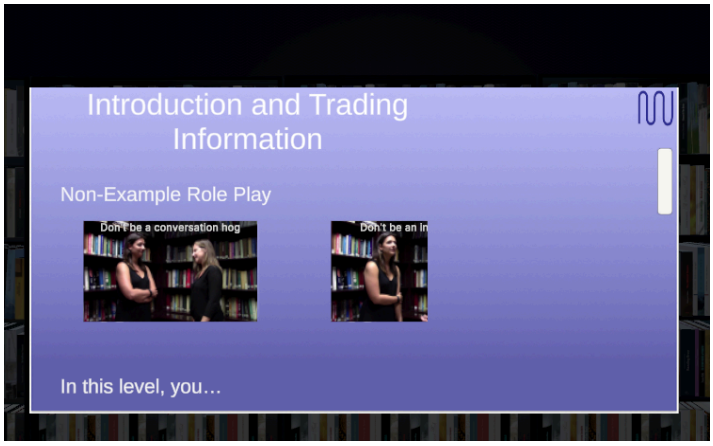


Figure 3 Lesson view: *The lesson displays the content of the specific lesson including videos and a general description.*

Summary:

When a user clicks on a 'Learn more' button for a lesson, they are directed to the lesson display that originally shows Non-example role play videos, a general description that can be expanded, and a button to see more content rules.

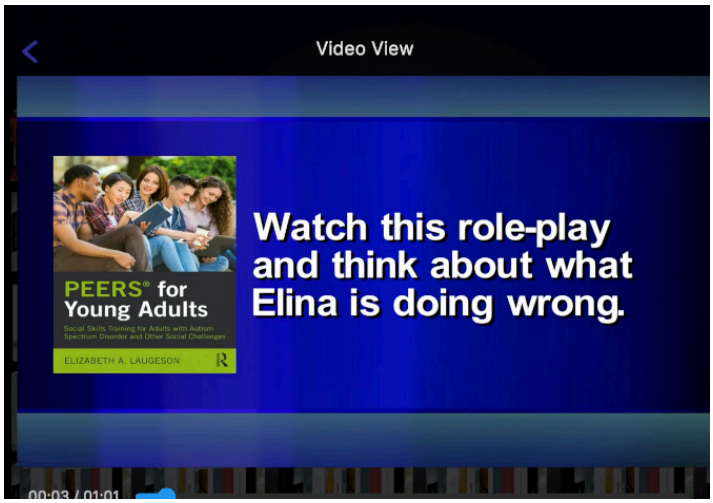


Figure 4 Video view: *The user can click on a video in the lesson view to play the video.*

Summary:

The user has a variety of videos that they can choose to watch being displayed on the lesson view. Once the user clicks the video they want to watch, they are taken to a separate view and the video begins to play.

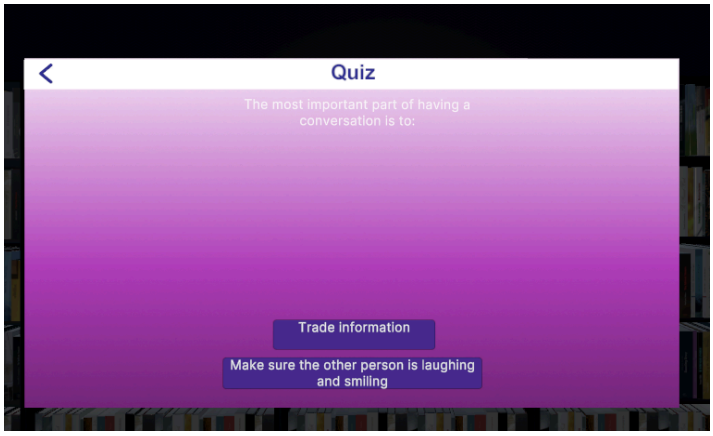


Figure 5 Quiz view: *When the user clicks on take quiz they will be taken to the quiz view and they are displayed with a prompt and two choices.*

Summary: The quiz view displays a question in white text at the top of the screen, and two purple boxes that have answers in white text.

5. Development, Testing, and Tools

In order to create this application, our team chose multiple different software and hardware tools for documentation and development. We were able to learn the skills necessary to use each tool, which allowed us to efficiently plan and complete this project to the best of our ability. Below is a description of each tool and program we used to aid us in our development process, and how each of them was used.

Github:

To track version control for our application, we used Github. All of our source code development was tracked and managed within a Github repository. To keep the size of the repository relatively small, only the Assets, Packages, and ProjectSettings directories were tracked by the repository. The project requires additional files to be opened as a Unity Project, but these files are automatically generated when the project is opened for the first time.

The "main" branch of this Github repository was reserved for stable releases. The "development" branch was for semi-stable development builds, such as versions of the application after each two week scrum. The "featureDev" branch was used for developing individual features, and this was where individual team members were able to write and push code to. Once the code was tested and approved, the new features in "featureDev" were merged into the "development" branch. Lastly, the "development" branch was merged with the "main" branch at major development milestones.

Unity:

The entire project was created as a Unity Project, and many of the tools we used were already included in the Unity editor. We modified many prefab assets from the VR core project template including interactable menus, the character controller, and the video screen. Additionally, for testing without a VR headset, we used Unity's XR Interaction Toolkit plugin.

Blender:

Our 3D library scene, which is set as the background room for this application, was created in Blender. Some aspects of the room were modified prefab assets and premade textures. After completion, the scene was exported into Unity.

Documentation:

All of our documentation and deliverables were created collaboratively using Google Docs. Additionally, all of our design documents are stored in a private Google Drive folder. We used Miro, a collaborative virtual workspace, to design diagrams for our deliverable documents. Finally, we used Figma, a collaborative interface design tool, to create our UI designs and mockups.

Hardware Tools:

We tested our project on both the Oculus Quest 1 and Oculus Quest 2, so our software is designed to run on both of them. Once we had a stable version of the software that was ready to be turned into an executable, we used the SideQuest desktop app in order to load that executable onto one of our headsets. Before we had a standalone executable to run on the headsets, we used a 2020 Razer Blade 15 laptop with Quest Link to run our software in Unity play mode.

Communication Tools:

We used Discord to facilitate our electronic communication and virtual meeting needs within our development team. For communication with our client, faculty advisor, and Dr. Elizabeth Laugeson, we used Gmail and Zoom.

Development Process:

Our development process was based on the spiral incremental strategy. At the start of the spring semester, we reviewed the project's core requirements and created an incremental development timeline. Every two weeks of the semester would be treated as an individual development sprint; with the goal of each sprint being to implement the next increment of features and functionality. At the end of each sprint we met to review our progress, discuss challenges, and plan for the next iteration.

6. Results

After a semester of planning and semester of development and testing, we were able to achieve the following results:

We created a fully functioning VR software that is usable and meets our clients requirements. The software will be used for research purposes as well as potential future development.

Our UI design was adapted from the original PEERS® mobile app, and we wanted to create something very similar that would work in a VR setting. We managed to create a visually appealing UI while maintaining the original UI of the PEERS® mobile app, but the goal was to make it more immersive and user friendly. Out of 18 lessons 5 are completed and fully implemented, the structure for all 18 lessons is currently implemented, but the content for those lessons is not. We programmed the connections to be able to dynamically load the lesson content from Json files, which makes the project efficient and scalable. The library scene we created added additional realism for the user.

However there were some limitations within the project, lesson view has some issues with scrolling and knowing the boundaries of the UI elements.When scrolling the content of the lesson view vertically, it is possible for the cursor to get stuck in the horizontally-scrolling content containing example and non-example videos which prevents the user from scrolling vertically. Links between objects were lost when creating an executable for the project

7. Future of PEERS® VR

This section will discuss the future of PEERS® VR and all the additional information needed if this project was to be continued by a future capstone group.

In the future, PEERS® VR is intended to be used for research by Dr. Howorth pertaining to the capabilities of VR when it comes to teaching social skills. Dr. Howorth may reach out to other developers through either research funds or future Capstone classes to continue developing PEERS® VR to a more finalized state before or after conducting research, as well. There are still several aspects of this application that have the potential to be further developed and additional features that we as a Capstone group simply did not have the time or resources to implement. One such notable feature is the inclusion of 360° videos to enhance the immersivity of the virtual reality learning environment. Any future extensions or additional features will depend on Dr. Howorth's desires and communication with UCLA for video content.

Due to an international situation with the original developers, the PEERS® mobile application may stop receiving updates in the future, so PEERS® VR has the potential to be used as a replacement.

7.1. Routine Tasks

For this project, there aren't many maintenance tasks that need to be done. If for any reason there is a bug during runtime it is recommended to restart the application. For more information on routine tasks and application upkeep, please refer to the Administrative Manual.

7.2. Periodic Administration

The PEERS® VR program does not require regular maintenance, but it is possible to expand the content of the program to include more of the PEERS® curriculum. There are currently 5 of the 18 lessons implemented. For more information on administrative tasks and maintenance, please refer to the Administrative Manual.

7.3. Additional Information

Future development will ultimately depend on Dr. Howorth's request, there were several features that were mentioned to us as potential desired additions to the application which we weren't able to get to. One such feature is text-to-speech. We hit a roadblock with the text-to-speech feature with Unity wherein a paid addon was necessary to achieve this feature, and most options were either monthly subscriptions or AI based. Finding a solution to the text-to-speech issue would increase the accessibility of the application as a whole.

With the inclusion of 360° videos, PEERS® VR would be able to take advantage of a higher level of realism and immersion. Adding these videos, however, would require filming them, efficiently storing the massive video files, and being able to dynamically load them within the application itself.

Another request from Dr. Howorth for PEERS® VR was to have the capability to store and track information for research purposes, such as how long the user spend watching the videos, how many lessons were viewed, how long was spent on each lesson, how many times they answered the quiz questions before passing the quiz, how many times the quiz was taken, etc. Dr. Howorth may be able to provide a more comprehensive list of desired statistics upon request.

Lastly, it was originally requested that PEERS® VR have a login functionality, allowing several users to save their information on the same headset. This feature would allow storage of individual players' data for research purposes, as well.

8. Conclusions

In conclusion, this document outlines a comprehensive review on the design and development process undertaken for our project, PEERS® VR, including all design elements, how each element advances overtime to fit into the final project, and how we went about the development process as a team. This document also includes a brief reflection on our process, including what we could have done differently, roadblocks we encountered, and what features were still left to add in the future.

As a group, we learned a lot about the process of software engineering from the beginning of the planning phase to the end of the development stage. We learned what it takes to communicate effectively with a client about their expectations and desires, how to research technology we had no previous experience with to the extent that we could use it in a project, independent problem-solving techniques, and how to work on code alongside other programmers while communicating and collaborating on every feature during the development process.

9. Acknowledgments

We would like to sincerely thank our client, Dr. Sarah Howorth, for proposing this project to us and trusting us to deliver a final product she could be proud of. This application wouldn't have been possible without her enthusiasm and dedication to such an incredible project. We would also like to thank our faculty advisor, Dr. Laura Gurney, for not only being such a wonderful capstone instructor, but for aiding us in our development journey and guiding us through this year-long process. We would also like to extend our gratitude to UCLA's Dr, Elizabeth Laugeson for allowing us to use the PEERS® curriculum and content. Lastly, we would like to thank our peer team, 5M (Michael Massari, Danny McA'Nulty, Madox Hussey, Nick Millett, and Mohammed

Fazli), for providing our team with feedback on both our documentation and the project itself throughout the duration of the course.

(Continued on next page)

References

JamTech.(2023) . Figma, https://www.figma.com/file/Iosmhnt2IlNk7ggfHybAt9/PEERS?type=design&node-id=0%3A1&mode=design&t=sEC0nLgmgZfr62xM-1

Laugeson E. A. (2014) The PEERS Curriculum for School-Based Professionals, Table 1.1 (1st Edition) https://www.routledge.com/The-PEERS-Curriculum-for-School-Based-Professionals-Social-Skills-Training/Laugeson/p/book/9780415626965

PEERS®. (2021). PEERS® (version 1.1.0) [Mobile app]. Apple Store OR Google Play. https://play.google.com/store/apps/details?id=com.peersclinic.peers&hl=en_US&gl=US

PEERS (2023) UCLA PEERS® Clinic, Semel Institute for Neuroscience and Human Behavior. https://www.semel.ucla.edu/peers

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). GitHub repository, https://github.com/VoloVita/PeersVR_Capstone/tree/main/Documentation/Deliverables

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). System Design Document (SDD), version 1.0

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). Systems Requirement Specification (SRS), version 2.0

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). User Interface Design Document (UIDD), version 1.0

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). Code Inspection Report (CIR), version 1.0

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). Administrator Manual (AM), version 1.0

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). User Guide (UG), version 1.0

Appendices

This section will include all the six major project artifacts (SRS, SDD, UIDD, CIR, AM, UG) that JamTech has completed. Each will also include a signature sheet affirming that everyone on the team contributed and approved the final product.

(Continued on next page)

A.1 Team Review Sign-off

This is the team review sign off meaning that all current team members of JamTech (Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud and Maha Fazli) have fully reviewed and agreed upon what has been implemented and documented to achieve a proof of concept for this project.

By typing one's name under the signature column and giving the date, the individual signs this document.

Name	Signature	Date
Allison Lupien	Allison Lupien	5/1/2024
Comments:		
Jacob Michaud	Jacob Michaud	5/1/2024
Comments:		
Maha Fazli	Maha Fazli	5/1/2024
Comments:		
Nick Sarno	Nick Sarno	5/1/2024
Comments:		
Tristan Cilley	Tristan Cilley	5/1/2024
Comments:		

A.2 Document Contributions

This is the current contribution of each team member towards the Project Report.

Name	% of contribution
Allison Lupien	20% [proof-reading and editing, organization and formatting, Requirements]
Jacob Michaud	20% [proof-reading, organization and formatting]
Maha Fazli	20% [proof-reading, organization and formatting]
Nick Sarno	20% [proof-reading, organization and formatting]
Tristan Cilley	20% [proof-reading, organization and formatting]

Appendix A - SRS

System Requirements Specification

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 2.0 approved

For Sarah K. Howorth

University of Maine

Nov 27, 2023

Prepared by JamTech: Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli

1. Introduction

This is a two-semester (**Fall 2023 - Spring 2024**) computer science capstone project to complete the requirements of a capstone experience at the University of Maine.

Dr. Sarah Howorth, is an associate professor of special education and program coordinator for the special education graduate programs at the University of Maine and she is also the director of the PEERS® Lab at UMaine and our primary client for this capstone project.

The PEERS curriculum was written by Dr. Elizabeth Laugeson. The curriculum has already been converted into a PEERS mobile application on iOS. The mobile app contains information, video role-play examples, and practice questions to help users learn social skills. Dr. Howorth is interested in creating the next evolution of the PEERS app. She believes that users can learn social skills more effectively through a VR interface as it provides a more private, immersive, and engaging experience.

This capstone project aims to create a proof-of-concept VR experience that can be used to solicit funding for further research.

1.1 Purpose of This Document

This document provides an initial overview of the entire project. Key areas of design such as functional requirements, non-functional requirements, use-case modeling, system diagrams, and acceptance testing shall be covered in greater detail. The primary audience of this document includes our client, Dr. Sarah Howorth, and ourselves, JamTech. The secondary audience includes the professor of our capstone course, Dr. Laura Gurney, and anyone interested in learning more about the project. Our intention is to provide a clear, detailed summarization of this project's requirements and to establish a schedule for upcoming deliverables.

1.2 Purpose of the Product

Dr. Sarah Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS curriculum was written by Dr. Elizabeth Laugeson. The curriculum has already been converted into a PEERS mobile application on iOS. The mobile app contains information, video role-play examples, and practice questions to help users learn social skills. Dr. Howorth is interested in creating the next evolution of the PEERS app. She believes that users can learn social skills more effectively through a VR interface as it provides a more private, immersive and engaging experience. This capstone project aims to create a proof of concept VR experience that can be used to solicit funding for further research.

1.3 Product Scope

The scope of the product, described by Figure 1, is to teach social skills i.e. (conversational skills, handling teasing and bullying). Additionally, the system is designed to be able to provide users with a virtual reality (VR) interface to be able to learn social skills in an engaging way. Figure 2 provides more information on how a user can interact with the system. The requirements of the system describe more precisely what the system will offer to the users. (***See section 2. Functional requirements****)*

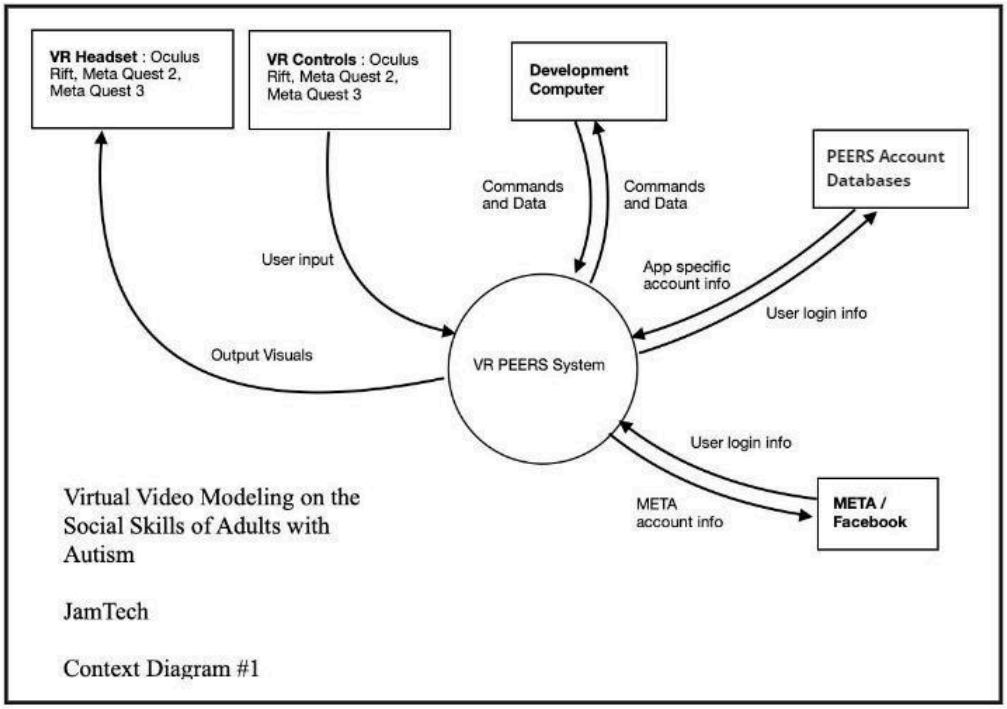


Figure B1 System Context Diagram: *The VR PEERS System will interact with META to log the user into the VR headset, with the PEERS databases to log the user in and get user information, with the*

development computer to transfer commands and data as well as the headset to display the scene and the controls to get user input.

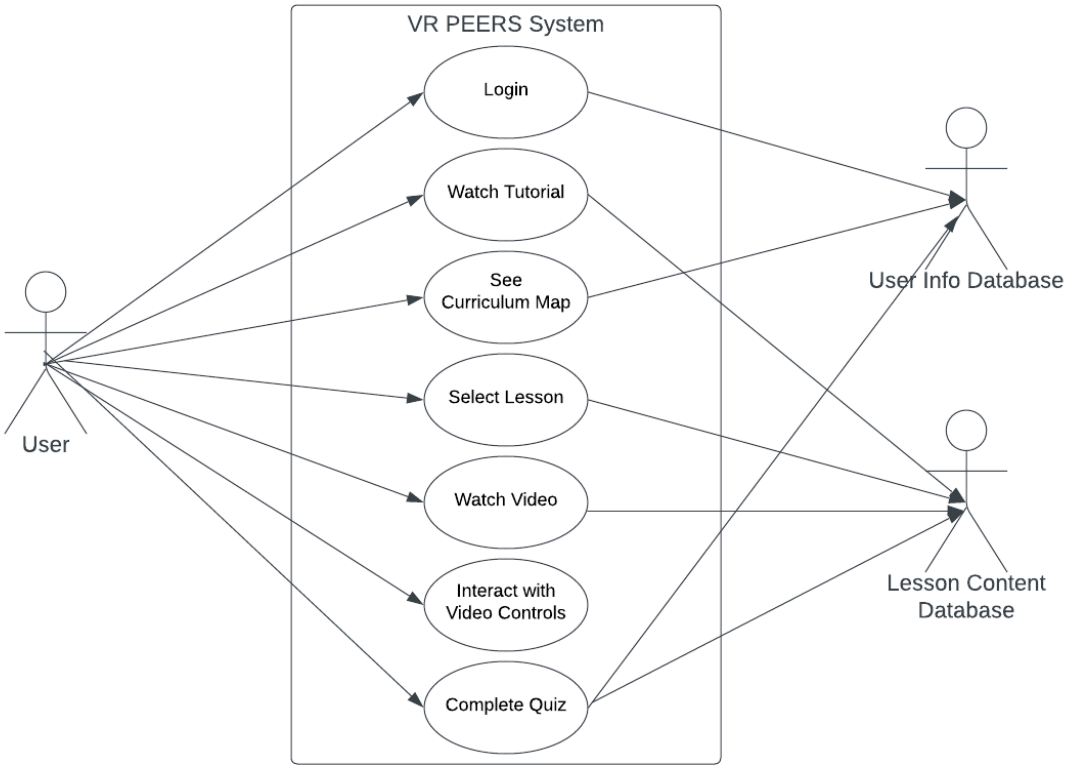


Figure B2 Top Level Use Case Diagram*: This use case diagram depicts the different ways a user would want to interact with the VR PEERS system as well as the additional actors that system interacts with, namely the User Info Database and the Lesson Content Database.*

There are 7 use cases depicted in Figure 2 Use Case Diagram. In general terms, a user of a system can log into the system, watch a tutorial, look at their curriculum map, select a lesson to learn from the map, watch lesson videos with standard media controls and complete lesson quizzes. Use cases take the form UC-# of the use case: Name - a use case statement. Each use case is reflected in the functional requirements. **(*See section 2. Functional requirements*)**

UC-1: Login - As a user, I want to be able to login to my PEERS account.
UC-2: See Curriculum Map - As a user, I want to be able to see my progress by viewing my curriculum map.
UC-3: Select Lesson - As a user, I want to be able to select a lesson to work on.
UC-4: Watch Video - As a user, I want to be able to watch lesson videos.

UC-5: Complete Quiz - As a user, I want to be able to complete a lesson quiz.
UC-6: Watch Tutorial - As a user, I want to be able to watch a tutorial video on how to use the system the first time I login.
UC-7: Interact with Video Controls - As a user, I want to be able to interact with standard media controls while playing a video.

2. Functional Requirements

The functional requirements below are written with the intended purpose of what the system will offer, as well as the requirements that the system will follow to provide an interface for the user to interact with. The functional requirements are each followed by a use-case table which displays detailed information about the requirement.

Functional Requirements follow the format: FR-(unique identification number): requirement description. The priority number for each functional requirement is included in its corresponding table. The priority number 1 represents the lowest priority and 5 the highest priority.

FR-1: The system shall allow users to log into their PEERs account by entering their email and password.

FR-2: The system displays the curriculum map to the user

FR-3: The system shall allow the user to select a lesson from the curriculum map.

FR-4: The system shall allow the user to view lesson-related video role-play examples in the VR environment.

FR-5: The system shall allow users to complete lesson assessments in the VR environment.

FR-6: The system shall offer a simple VR tutorial and safety warnings to new users.

FR-7: The system shall allow standard media controls for the content videos.

FR-1: The system shall allow users to log into their PEERs account by entering their email and password.

Number	1
Name	Log into PEERs account

Summary	Allow a user to enter an email and password to log into their PEERs account	
Priority	5	
Preconditions	User must have a PEERs account	
Postconditions	User is logged in and is granted access to the rest of the application's functions	
Primary Actor	User	
Secondary Actors	Account Database	
Trigger	User selects the "Login" button	
Main Scenario	**Step**	**Action**
	1	A log-in window is displayed
	2	User enters their email and password into the text boxes
	3	User clicks the "Login" button to submit their information
	4	User is logged in
	5	User is brought to curriculum map
Extensions	**Step**	**Branching Action**
	4a	The information entered by the user does not match an account: ● A pop-up message alerts the user that the information entered does not match an existing account, prompting them to try again
	5a	The user is new: ● Instead of being brought to the curriculum map, the user is shown the tutorial videos first.

FR-2: The system displays the curriculum map to the user

Number	2	
Name	Display Curriculum Map	
Summary	Allows the user to view the curriculum map	
Priority	5	
Preconditions	User must be logged in	
Postconditions	User is presented with their current curriculum map	
Primary Actor	User	
Secondary Actors	Curriculum database	
Trigger	User logs in successfully, or User backs out of a lesson	
Main Scenario	**Step**	**Action**
	1	System queries database for user’s current curriculum map
	2	User is shown the curriculum map

FR-3: The system shall allow the user to select a lesson from the curriculum map.

Number	3	
Name	Lesson Selection from Curriculum Map	
Summary	Allow the user to select a specific lesson from their curriculum map to enter the ‘lesson overview’ mode.	
Priority	4	
Preconditions	User must be logged in. User is on the Curriculum Map view.	
Postconditions	User is able to view the role-play videos within the lesson	
Primary Actor	User	
Secondary Actors	Curriculum database	
Trigger	User selects a lesson from the curriculum map.	
Main Scenario	**Step**	**Action**
	1	User selects a specific unlocked lesson: lesson overview mode is entered
	2	The lesson overview is displayed
Extensions	**Step**	**Branching Action**
	1a	User selects a specific locked lesson on the curriculum map: ● A pop-up window displays a message alerting the user that they have not yet unlocked this lesson

FR-4: The system shall allow the user to view lesson-related video role-play examples in the VR environment.

Number	4	
Name	Viewing Role-play Videos	
Summary	User has selected a lesson, and now they are presented with a selection of videos from which the user can select to watch.	
Priority	3	
Preconditions	User is currently in ‘lesson overview’ mode	
Postconditions	Once all video content has been viewed, the VR quiz is unlocked.	
Primary Actor	User	
Secondary Actors	VR Peers System	
Trigger	User enters ‘lesson overview’ mode	
Main Scenario	**Step**	**Action**
	1	User selects a video
	2	The systems enters the 3D video scene
	3	The user watches the content of the video
	4	The user returns to the lesson display

FR-5: The system shall allow users to complete lesson assessments in the VR environment.

Number	5
Name	Complete Lesson Assessment
Summary	Allow a user to complete a lesson assessment (quiz)

Priority	4	
Preconditions	User must have completed a video lesson material and be in the lesson overview mode	
Postconditions	Lesson marked complete and progress is reflected in the curriculum map	
Primary Actor	User	
Secondary Actors	Curriculum Database	
Trigger	User clicks “Take the quiz” button	
Main Scenario	**Step**	**Action**
	1	Display VR immersive video
	2	Video pauses at the end
	3	Simple binary choice quiz is displayed for the user
	4	User selects answer
	5	Correct response video is displayed
	6	Assessment ends and user is returned to the lesson overview mode
Extensions	**Step**	**Branching Action**
	5a	If incorrect response is chosen, incorrect response video is displayed
	5b	User is returned to step 3

FR-6: The system shall offer a simple VR tutorial and safety warnings to new users.

Number	6	
Name	Display tutorial and safety warnings	
Summary	The system displays a simple tutorial and safety warnings to new users after logging in for the first time.	
Priority	5	
Preconditions	New user must be logged in	
Postconditions	User is brought to the curriculum map	
Primary Actor	User	
Secondary Actors	Account Database	
Trigger	New user successfully logs in	
Main Scenario	**Step**	**Action**
	1	The system displays safety warnings for the user to read before proceeding
	2	User clicks the “Continue” button
	3	The system displays a tutorial for the user to watch before proceeding
	4	User clicks the “Continue” button

FR-7: The system shall allow standard media controls for the content videos.

Number	7	
Name	Media controls	
Summary	When the system displays content videos, the media controls should follow standard conventions	
Priority	2	
Preconditions	User has selected a video	
Postconditions	User is able to interact with content videos	
Primary Actor	User	
Secondary Actors	Media content	
Trigger	Content video is played	
Main Scenario	**Step**	**Action**
	1	User clicks the video to bring up the media controls
	2	User interacts with the video using standard media controls
	3	The corresponding media control response is applied to the video
	4	The video ends
Extensions	**Step**	**Branching Action**
	1a	The user does not click the video: ● Nothing happens and the video plays through uninterrupted.

2.1 Acceptance Testing

The following acceptance tests are created with the intention of thoroughly testing all of the functional requirements listed above. Each acceptance test will test one or multiple of the functional requirements, ensuring that every aspect of the program will function as expected.

The acceptance tests will be written in the following format:
Acceptance Test #ID (list of all functional requirements that will be tested):
Overview
[In depth description]

Acceptance Test 1 (1, 6, 7):
The user logs into the application using their PEERs account as a new user.
This test will ensure that users are able to successfully log into an account while also ensuring that new users will be shown safety warnings and tutorial videos upon logging in. This will also test if the standard media controls are functioning for the tutorial videos.

Acceptance Test 2 (2):
The user opens the curriculum map.
This test will ensure that the curriculum map functions and displays properly.

Acceptance Test 3 (1, 2, 6):
The user logs into the application as an existing user.
This test will ensure that users are able to successfully log into an account that has tracked their progress from previous sessions. This will also ensure that the safety warnings and tutorial videos are only displayed for new users and not existing videos. Aside from this, logging in as an existing user and checking the curriculum map will confirm whether the map has tracked the user's previous progress or not.

Acceptance Test 4 (2, 3):

The user chooses a lesson.

This test will ensure that the user is able to move between the curriculum map and the lesson overview mode by selecting a lesson from the curriculum map. This will also ensure that the lesson overview mode displays as expected.

Acceptance Test 5 (3, 4, 5, 7):

The user completes a lesson.

This test will ensure that the user is able to enter a new lesson and complete it while testing all of the features associated with completing a lesson. First, it will ensure that the role-play videos display properly. This test will also ensure that content videos can be interacted with using standard media controls. As for the assessment portion, this test will ensure that the assessment environment functions as expected, and that the user is able to answer the assessment questions as expected.

Acceptance Test 6 (5):

The user intentionally selects the wrong answer in a role-play scenario.

This test will ensure that the program proceeds the way it's intended to and plays the correct corresponding videos when the incorrect answer is selected.

Acceptance Test 7 (2, 3, 5):

The user exits the lesson overview to return to the curriculum map after completing an assessment.

This test will make sure the progress from completing the assessment is properly saved, and that the assessment ends when it's supposed to. This will also test to ensure that the lesson overview mode still displays properly after the assessment, and that the user is able to return to the updated curriculum map from the lesson overview mode.

3. Non-Functional Requirements

The non-functional requirements (**NFR**) are meant to display requirements that are related to the characteristics of the system. NFRs describe constraints and capabilities of the system (performance requirements, safety and privacy requirements, security requirements, and software quality attributes).

Non-Functional Requirements follow the format: NFR-unique identification number: requirement description (Priority number). The priority number 1 represents the lowest priority and 5 the highest priority.

3.1 Performance Requirements

NFR-1: The system shall display the lesson to the VR headset within 0.5 seconds, 99% of the time. (Priority 5)

NFR-2: The system shall accurately keep track of the user's curriculum progress 99% of the time. (Priority 3)

NFR-3: The system shall be able to handle up to 100 active users 95% of the time. (Priority 4)

NFR-4: The system shall be able to handle 100 login requests every 5 seconds 95% of the time. (Priority 3)

NFR-5: The system shall be able to handle the user input within 0.5 seconds 95% of the time. (Priority 4)

NFR-6: The system shall be able to update based on user actions within 0.5 seconds 95% of the time. (Priority 4)

3.2 Safety and Privacy Requirements

NFR-7: The system shall comply with the California Consumer Privacy Act (California) (Priority 4)

NFR-8: The system shall comply with the General Data Protection Regulation (European Union) (Priority 4)

NFR-9: The system shall protect user's data from unauthorized access. (Priority 4)

3.3 Security Requirements

NFR-10: The system shall be able to protect user data from attacks 99% of the time. (Priority 4)

3.4 Software Quality Attributes

NFR-11: The system shall be able to connect to internet services 95% of the time. (Priority 3)

NFR-12: The system shall be available to all users 24 hours a day, 95% of the time. (Priority 3)

NFR-13: The system shall not exceed a size of 20GB. (Priority 1)

4. User Interface

See "User Interface Design Document (**UIDD**) for Virtual Video Modeling on the Social Skills of Adults with Autism" for user interface designs. Estimated availability on December 4th, 2023.

5. Deliverables

The deliverables for this project will be maintained on our central GitHub repository. Physical copies of important documents will be provided in person. In the section below, all deliverables are separated by the semester in which they will be first published. Almost all deliverables will be subject to updates and modifications over the entire year. Our GitHub repository will contain the most current versions of all deliverables; additional physical copies will be provided as requested.

Deliverables for Fall 2023 - [COS397]

Hard copies of each of the following:

- Systems Requirement Specification
- System Design Document
- User Interface Design Document

An electronic file containing the following:

- Systems Requirement Specification **(available: Nov 1, 2023)**
- System Design Document **(available: Nov 15, 2023)**
- User Interface Design Document **(available: Nov 29, 2023)**

Deliverables for Spring 2024 - [COS497]

Hard copies of each of the following:

- User Manual
- Administrator Manual

An electronic file containing the following:

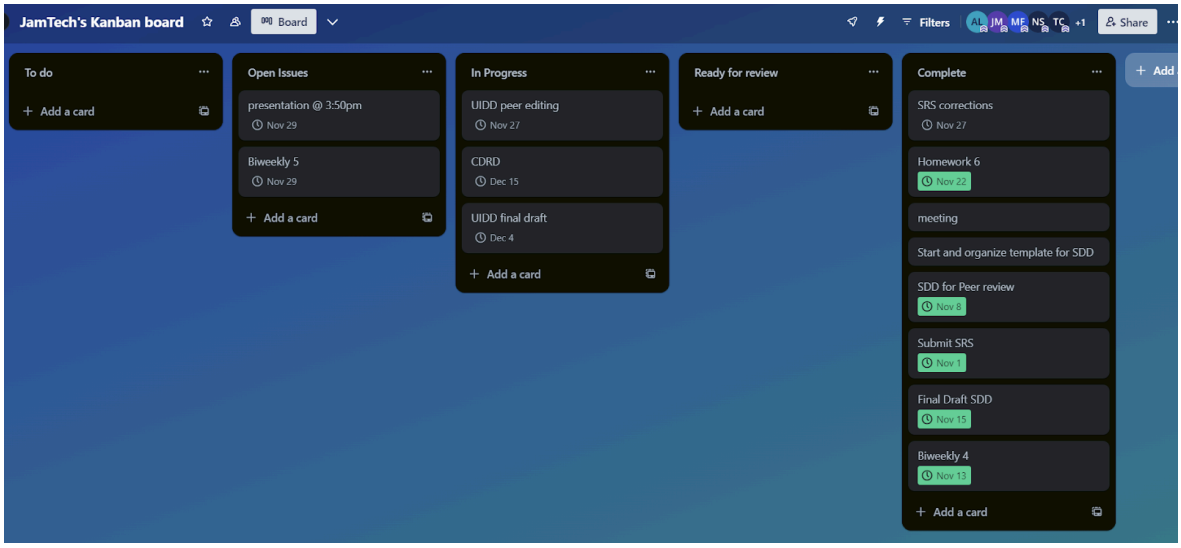
** Documents hosted on Github will be updated regularly **

- Critical Design Review Document **(Estimated availability: to be announced)**
- Code Inspection Report **(Estimated availability: to be announced)**
- Final Project Report **(Estimated availability: to be announced)**
- User Manual **(Estimated availability: to be announced)**
- Administrator Manual **(Estimated availability: to be announced)**
- All source code **(Estimated availability: to be announced)**
- The executable program **(Estimated availability: to be announced)**
- Any other software required for installation and execution of the delivered program. **(Estimated availability: to be announced)**

6. Open Issues

This section will cover any issues that are currently open that have not reached a conclusion, a detailed description of the issue will be available. The issues will be addressed by their target completion date.

Open issues will be tracked in our kanban board:
https://trello.com/invite/b/erGa4KSJ/ATTI3a00e58a77e188a54783a2636c467fb372962312/jamtechs-kanban-board



Appendix B - SDD

System Design Document

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

Nov 15, 2023

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno,

Jacob Michaud, Maha Fazli

1 . Introduction

This is a two-semester (**Fall 2023 - Spring 2024**) computer science capstone project to complete the requirements of a capstone experience at the University of Maine. This is a project for Dr. Sarah Howorth, on virtual video modeling of the social skills of adults with autism.

Dr. Sarah Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS curriculum was written by Dr. Elizabeth Laugeson. The curriculum has already been converted into a PEERS mobile application on iOS. The mobile app contains information, video role-play examples, and practice questions to help users learn social skills. Dr. Howorth is interested in creating the next evolution of the PEERS app. She believes that users can learn social skills more effectively through a VR interface as it provides a more private, immersive, and engaging experience. This capstone project aims to create a proof-of-concept VR experience that can be used to solicit funding for further research.

1.1 Purpose of This Document

The purpose of this document is to describe our project's system design and architecture. This document is intended to both guide new users or laymen as well as serve as an official design document for the project. It will cover topics including system architecture, data-flows, file structures, and databases. The primary audience of this document includes our client, Dr. Sarah Howorth, and ourselves, JamTech. The secondary audience includes the professor of our capstone course, Dr. Laura Gurney, and anyone interested in learning more about the project.

2 . System Architecture

This section of the system design document will cover the system's architecture, which refers to how the system is planned to be built. The system's architecture will also refer to the logical architecture of the program which is the structure of the system and decomposition description. ***See 1.2 references for link to all diagrams.***

2.1 Architectural Design

The architectural design of the system is the logical architecture of a system, which can be summarized as how the system is planned to be built. The diagrams below will represent the logical architecture of the system followed by a description for each.

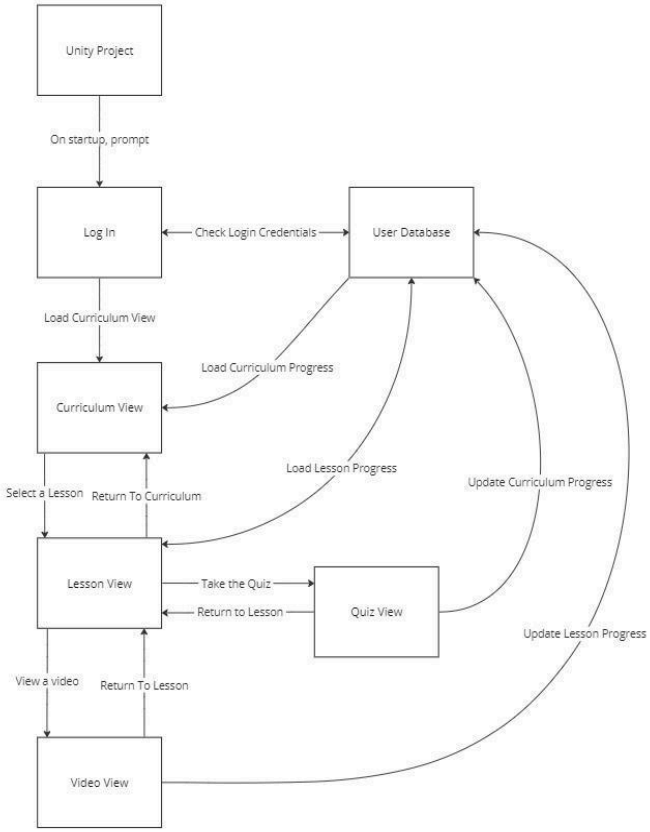


FIG C1.0 Software Architecture/Flow Diagram*: This graph showcases how the software parts interact with each other with the individual elements represented by boxes and arrows indicating interaction actions connecting everything.*

Our system flow diagram represents the relationship between different aspects of the software and their interactions with each other. The arrows are labeled with the corresponding functionality that occurs between each object. The direction of the arrow indicates the flow of information from one object to another. At most, the objects' points of reference are compared with the data within the user database in order to display the correct corresponding information to the user.

A basic data flow example using our diagram would look like this: on startup, the user is prompted to login to the system. From there, their login information is compared to that in our system’s user database and if correct, the curriculum view is displayed to the user. The user can then decide to select a lesson which will bring them into the lesson view which displays a selection of videos to watch. The user can view a video which after completion will update the

lesson progress of that user and return them to the lesson view. Upon completion of all of the videos in a given lesson, the user can enter the quiz view and upon completion of the quiz the curriculum progress will be updated and returned to the curriculum view.

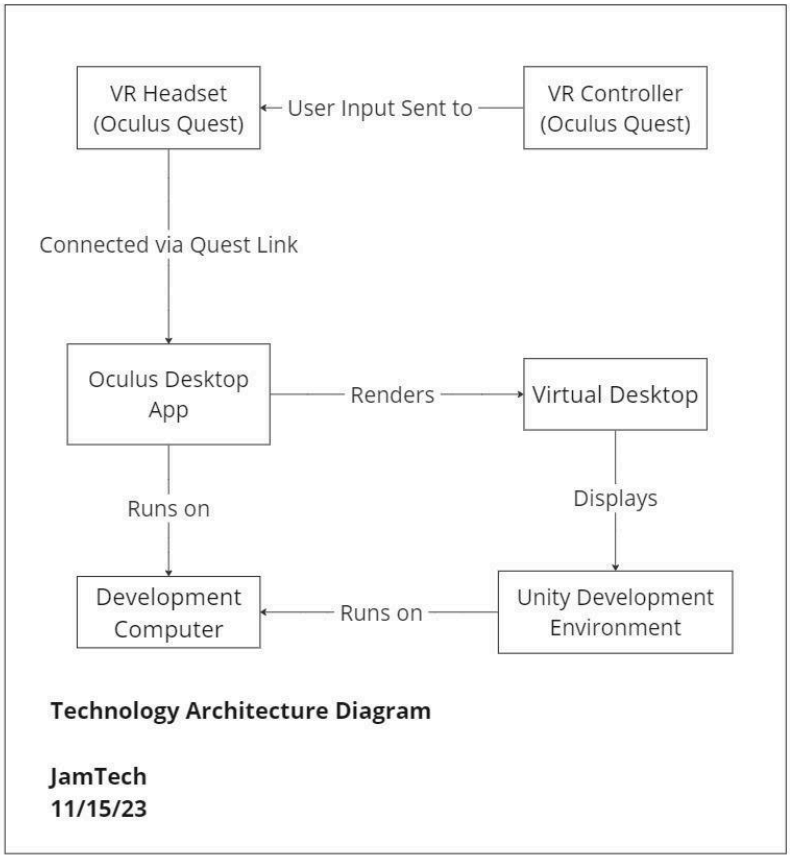


FIG C2.0 Technology Architecture Diagram*: This diagram shows the Oculus Quest connecting to our development computer via Quest Link. Once connected, we can view the Unity Development Environment in the VR headset and interact with Unity via the controllers.*

Fig C2.0 depicts the interactions between our hardware and software components. The VR equipment connects to the development computer via Quest Link which requires a USB-C 3.0 cable. Once connected, a virtual desktop environment can be accessed through the Oculus

Desktop Application. The user can open a Unity Development Environment in the virtual desktop, and run the project. When played, the project is rendered in the VR headset and Unity receives input via the controllers.

2.2 Decomposition Description

This section will describe the decomposition of the system presented in section 2.1, the diagram provided will illustrate the system's design.

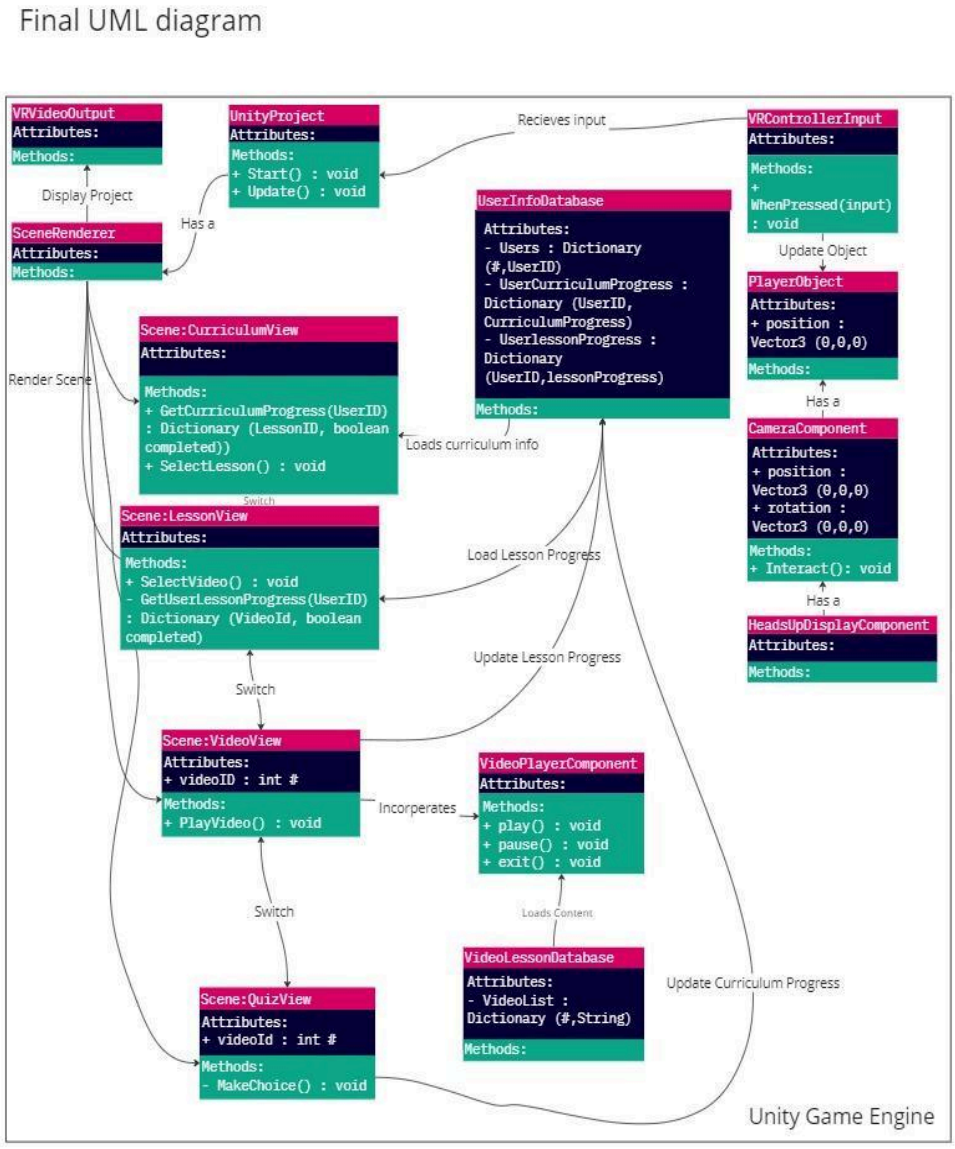


FIG C3.0 Unity Game Engine*: This diagram depicts the system's architectural design, represented by core components, their attributes, and their methods. The boxes contain this*

information and are connected by arrows with action words to describe how each component interacts with the system and the other components.

The current system architecture can be represented by a context diagram and a UML diagram. The core component of this system is the Unity Project, which is the main software component represented by the UML diagram. The Unity Project is hosted on a desktop computer that has VR-ready GPU processing capabilities. The UML diagram contains multiple components that display the software's logic, including but not limited to: user interactions, video playback, logic within the application, and video/scene rendering. There are also multiple databases that are used to store information about users and their curriculum, lesson, and video progress.

Within the UML diagram are arrows indicating the interactions between different components of both the software and hardware. For example, the VRControllerInput interfaces with the unity project, and has a direct correlation to the PlayerObject, which will update the world and current view of the software to be displayed via the VRVideoOutput. In a simpler context, the desktop computer serves as the host of the application, the VR headset and controller are used to interact with the application, and the software and databases process the information needed to transfer the data between multiple systems.

3 . Persistent Data Design

This section outlines how the system will store data for this project; it is broken into sections 3.1 and 3.2.

Section 3.1 describes the databases used in this project and how they store data using entity-relationship diagrams. After each diagram there is a table that elaborates each element with its data-type, size and a brief description. Any personal identifiable information will be stored as a hash rather than plaintext. We intend to use NoSQL for the User Info and Lesson Content databases. We plan to use MongoDB to facilitate this design choice.

Section 3.2 includes a diagram of the GitHub repository for this project and a basic file structure for the Unity project.

3.1 Database Descriptions

The following diagram represents the configuration of our project's databases and their respective data. This project will include two separate databases. Figure C4.0 is a diagram of the User Info

Database, which will store the user's information like username, name, password, curriculum progress, ect. Figure C5.0 is a diagram of the Lesson Content Database that includes lesson content like videos and quizzes.

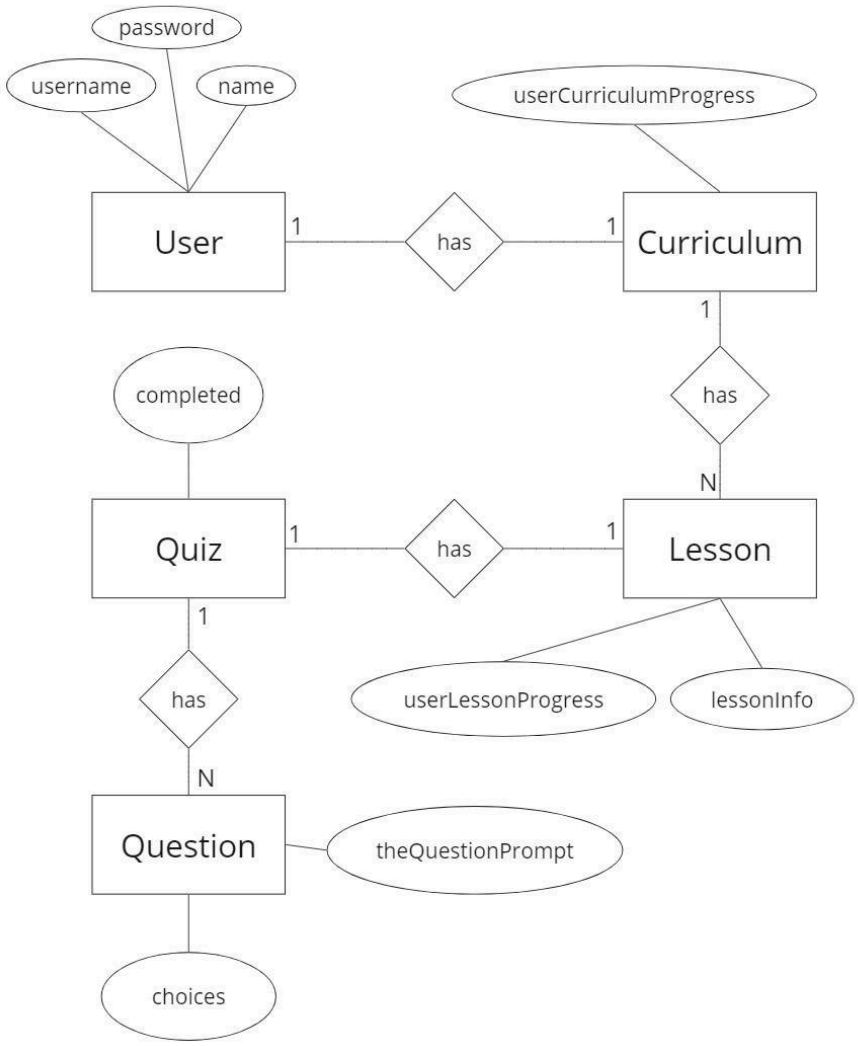


FIG C4.0 User Info Database Diagram*: The diagram above shows the layout of the User Info Database. Objects are represented by boxes, and the data those objects possess are represented by ovals. The lines represent a relationship between two components while the number or 'N' represent the ratio between objects. Object to data ratio is always one to one so a number is not used.*

Table C1: User Info Database Table

Table C1 provides more detail for each element seen in Figure C4.0 above.

Name	Data-type	Size	Description
User	object	varies	Holds the information username, password, name, and a Curriculum object that pertains to a specific user
username	string	64 bytes	The username of a user
password	hash value	16 bytes	The password of a user
name	string	35 bytes	The preferred first name of a user
Curriculum	object	varies	Holds the information userCurriculumProgress and has a Lesson object for every lesson in the PEERS curriculum
userCurriculumProgress	int	4 bytes	Indicates which lesson number the user is currently on
Lesson	object	varies	Holds in information lessonInfo, userLessonProgress, and a Quiz object
lessonInfo	string	varies	The link to the lesson data in the Lesson Content Database
userLessonProgress	dictionary	varies	Holds the information of whether a user has watched each video of a lesson. The key is the video ID and the value is a boolean representing if the video has been watched or not.
Quiz	object	varies	Holds the information completed and one or more Question objects
completed	boolean	1 bit	Indicated whether quiz has been completed or not

Question	object	varies	Holds the information theQuestionPrompt and choices
theQuestionPrompt	string	varies	The link to the question prompt in the Lesson Content Database
choices	string	varies	The link to the choices of the question in the Lesson Content Database

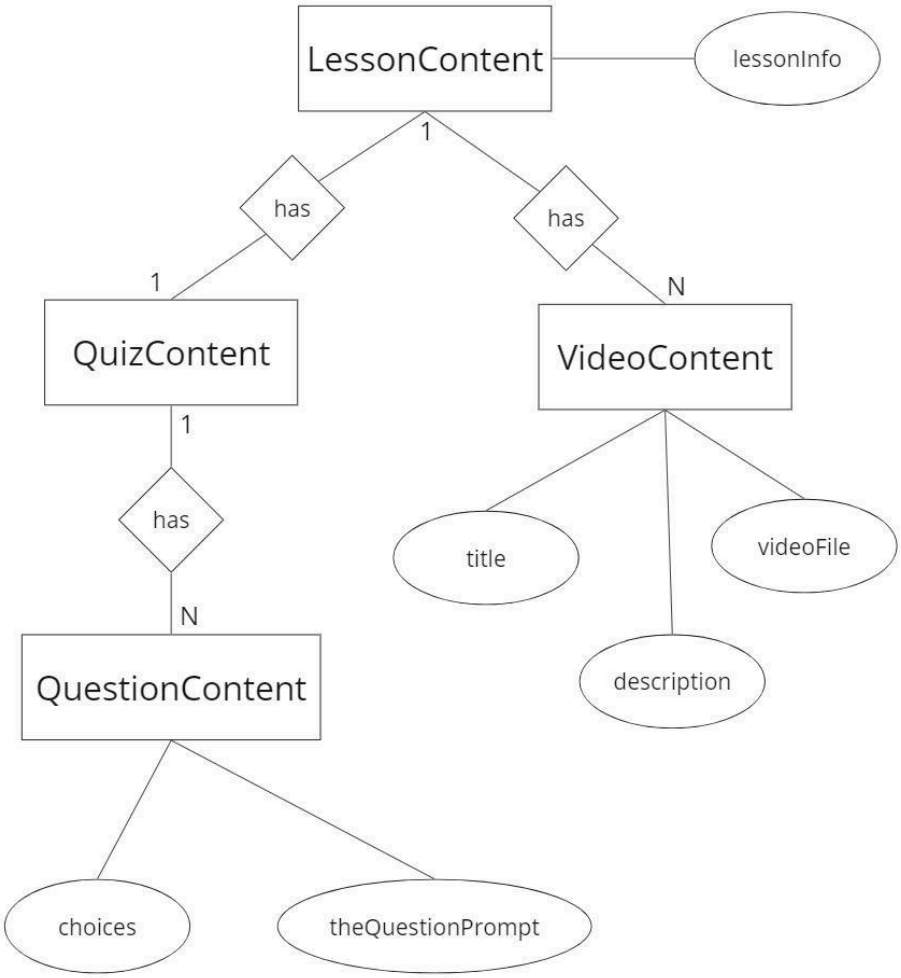


FIG C5.0 Lesson Content Database Diagram: The diagram above shows the layout of the Lesson Content Database.*Objects are represented by boxes, and the data those objects possess*

are represented by ovals. The lines represent a relationship between two components while the number or 'N' represent the ratio between objects. Object to data ratio is always one to one so a number is not used.

Table C2: Lesson Content Database Table.

Table C2 provides more detail for each element seen in Figure C5.0 above.

Name	**Data-type**	**Size**	**Description**
LessonContent	object	varies	Holds the lessonInfo file, multiple VideoContent, and one QuizContent
lessonInfo	txt file	varies	Holds additional lesson information from the PEERS curriculum if necessary
VideoContent	object	varies	Holds the title, description and the video file for a specific video
title	string	varies	The title of the video
description	string	varies	A brief description of the video
videoFile	video file	varies	The video file
QuizContent	object	varies	Holds multiple QuestionContent
QuestionContent	object	varies	Holds theQuestionPrompt and the choices for a specific question
theQuestionPrompt	string	varies	The test of the question
choices	dictionary	varies	The choices are accessed by a key and holds both the string text of the choice and a boolean that indicates if the choice is the correct or incorrect answer to the pertaining question

3.2 File Descriptions

This section includes a diagram of the system's file structure followed by a detailed table describing the name, data type, and size of each and every file. Our GitHub repository holds files that are relevant to the structure of the repository, electronic records of our deliverables, a detailed inventory of our VR hardware and a directory for everything specific to our Unity project.

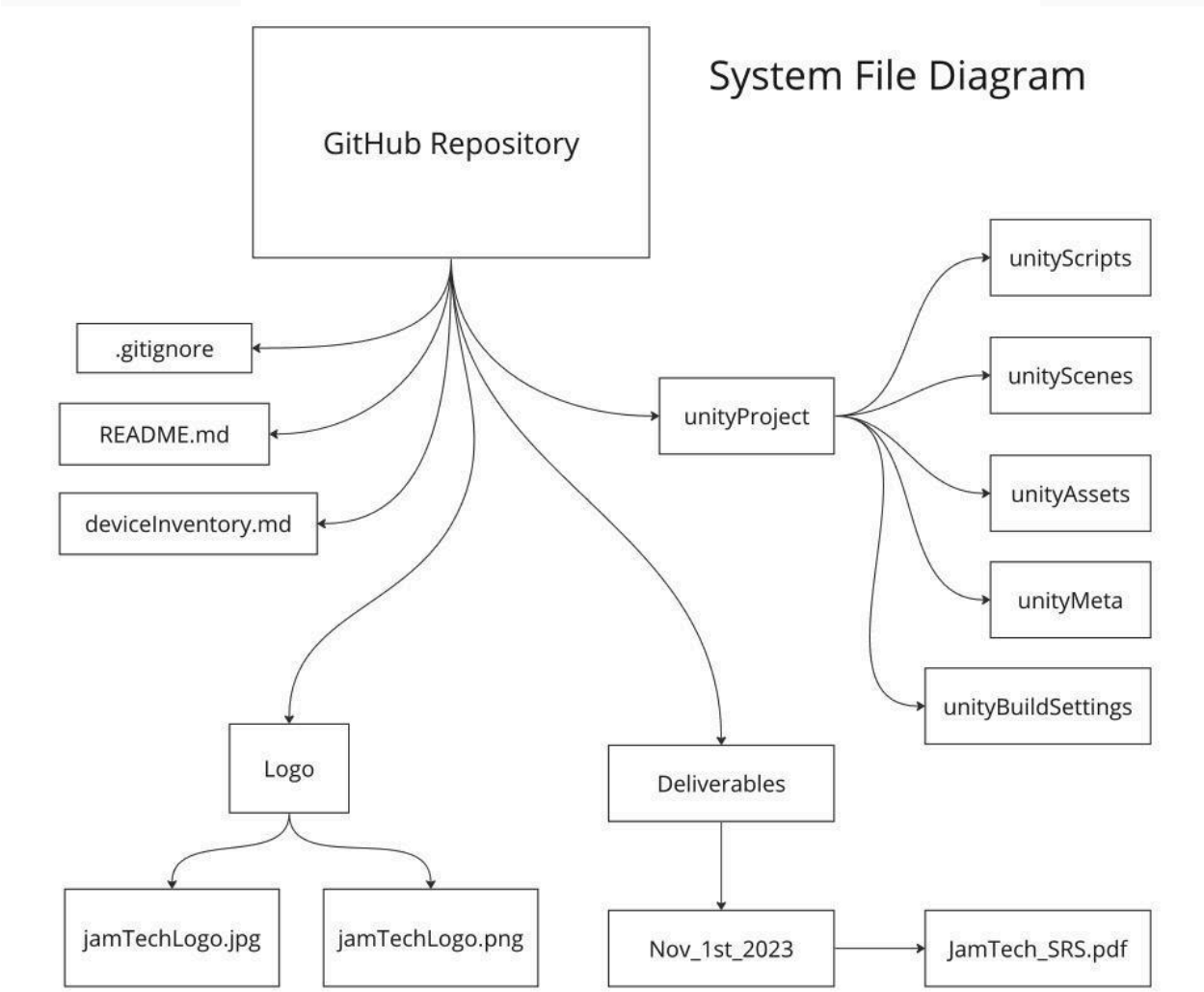


FIG C6.0 System File Diagram: The diagram above shows our system's file structure, which will be available in Github. The diagram shows the various folders and files represented by boxes, with arrows indicating what folders will contain which documents.

Table C3: File System Table

Table C3 provides more detail for each element seen in Figure C6.0 above.

File name	Data-type	Size	Description
README.md	Markdown Text	1 KB	An introduction to our team and the project overall
deviceInventoryVR.md	Markdown Text	1KB	A detailed inventory of our VR equipment
Logo	Directory	53 KB	Contains .jpg and .png images of team logo for use in documentation
Deliverables	Directory	400 KB	Contains .pdf versions of our deliverables and documentation.
.gitignore	Git Preferences	0.5 KB	Detailed list of files not to be included in our GitHub Repository
Unity Scripts	C# code	TBD	Custom scripts written for our Unity project
Scene & Asset Files	Various	TBD	Unity uses scene files and various asset files to define the game world, characters, textures, models, and other game assets
Project Settings	Text files	TBD	Unity's project settings are stored as text files in the ProjectSettings folder.
.meta files	Unity Specific	TBD	Unity generates .meta files for each asset to maintain metadata and connections between assets.
Build Settings	Unity Specific	TBD	Any custom build settings or build configurations that are critical to our project

4 . Requirements Matrix

This section refers to the functional requirements discussed in section 2 of the SRS (system requirements specification) and this matrix is represented in a tabular format to refer the functional requirements to its systems components.

Table C4: Functional Requirements to System Components

Table C4 references each functional requirement as developed in the SRS document and provides the name of the requirement, a brief description and lists all related system components.

ID	Name	Description	System components
FR-1	Log into PEERs account	Allow a user to enter an email and password to log into their PEERs account	UserInfoDatabase
FR-2	Display curriculum map	Allows the user to view the curriculum map	GetCurriculumProgress(UserID)
FR-3	Lesson selection from curriculum map	Allow the user to select a specific lesson from their curriculum map to enter the ‘lesson overview’ mode	SelectLesson( )
FR-4	Viewing role-play videos	User has selected a lesson, and now they are presented with a selection of videos from which the user can select to watch	PlayVideo( )
FR-5	Complete lesson assessment	Allow a user to complete a lesson assessment (quiz)	MakeChoice( )
FR-6	Display tutorial and safety warning	The system displays a simple tutorial and safety warnings to new users after logging in for the first time	HeadsUpDisplayComponent
FR-7	Media controls	When the system displays content videos, the media controls should follow standard conventions	HeadsUpDisplayComponent

Appendix C - UIDD

User Interface Design Document

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

Dec 4, 2023

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli

1 . Introduction

This is a two-semester (**Fall 2023 - Spring 2024**) computer science capstone project to complete the requirements of a capstone experience at the University of Maine. This is a project for Dr. Sarah Howorth on virtual video modeling of the social skills of adults with autism. Dr. Howorth is the director of the PEERS® Lab at UMaine, and our primary client for this capstone project. The PEERS® curriculum was written by Dr. Elizabeth Laugeson.

The following description comes from the website for the UCLA PEERS® Clinic,

> "The Program for the Education and Enrichment of Relational Skills (PEERS®) is world-renowned for providing evidence-based social skills treatment to preschoolers, adolescents, and young adults with autism spectrum disorder (ASD), attention deficit/hyperactivity disorder (ADHD), anxiety, depression, and other socio-emotional problems." (PEERS®, 2023)

The curriculum has already been converted into a PEERS® mobile application on iOS. The mobile app contains information, video role-play examples, and practice questions to help users learn social skills. Dr. Howorth is interested in creating the next evolution of the PEERS® app. She believes that users can learn social skills (eg; communication, humor, dating etiquette, etc) more effectively through a VR interface as it provides a more private, immersive and engaging experience. This capstone project aims to create a proof-of-concept VR experience that can be used to solicit funding for further research.

1.1 Purpose of This Document

This document shall serve as a guide to the application's user interface and visual design. The document will cover the following topics: user interface standards, user interface walkthrough, and data validation. The primary audience of this document includes our client, Dr. Sarah Howorth, and ourselves, JamTech. The secondary audience includes the professor of our capstone course, Dr. Laura Gurney, and anyone interested in learning more about the project.

2 . User Interface Standards

This section focuses on the standards chosen for the design of this application, encompassing logical choices such as layout and design principles, common components like menus and screens, and visual aspects including colors and fonts. These standards will be consistently applied throughout the user interface design of the entire application.

To see the flow from one screen to another, please see ***Figure D1.0*** in the JamTech SDD, ***Software Architecture/Flow Diagram***, which details the different screen areas. For visual aids for each of the screen areas, as well as explanations of each, please see ***Section 3: User Interface Walkthrough***, below.

Logical Choices:

1. Keeping titles and pop up displays centered.
2. The curriculum map is on the home screen because the curriculum map in the original PEERS® app was also the home screen, and because it's the largest navigation menu in the program. The curriculum map will be horizontal because there is more horizontal space in VR.
3. The screens will not be cluttered in order to avoid overstimulating the user, especially because it's easier to overstimulate users with a crowded screen in virtual reality, so we took that into account.

Common Components:

1. The white bar at the top with the title will remain the same, except the title will change as needed.
    a. The map icon will be there if the user is not in the curriculum map to return the user to that screen. This was taken from the original PEERS® app.
4. All of the lesson layouts will be the same
    a. Popup window prompting for more information, video carousel, questions in the quizzes, etc.

Visual Aspects:

1. The default background color for the application will be [Linear gradient (FFFFFF, B03EBA, 5F2065)] and the color for the buttons will be [BF3CAA].
2. Each lesson in the curriculum map will be assigned a unique color.
3. Within the lessons, the background color will vary depending on the color assigned to the lesson being viewed
4. Within the lessons, the color for the buttons will vary depending on the color assigned to the lesson being viewed.
5. The pop-up windows displayed when viewing lessons on the curriculum map will be [Linear gradient ((7D7EF0,2F2F94)]
6. The default text color throughout the program will be white
    a. The text in the login input boxes will be gray for contrast
    b. The text in the screen titles will match the pop-up windows when viewing lessons.
7. We will use the font [San-serif font "Inter"] for all of the text.
8. The map icon to return to the curriculum map in the top left-hand corner of the screen will be shaped like a zig-zag to match the design shape of the curriculum map that shows on the curriculum map screen.

9. The curriculum map will be shaped in a zig-zag to make better use of the space, and also in reference to the curriculum map designed in the original PEERS® app.

2.1 Accessibility and Inclusive Design Elements

Our application design prioritizes accessibility and inclusivity through diverse video content and by addressing the needs of users with visual, hearing, physical, or mental impairments. The following section outlines the features we intend to implement to meet our goals of accessibility and inclusion.

Visual impairment: (dyslexia / color blindness)

1. High contrast colors will be utilized throughout the project, especially with text against a background with color.
2. Color blindness mode
3. Text-to-Speech will be available for textual information such that a screen reader would be able to be used

Hearing impairment:

1. Video will have high quality clear audio
2. The user will be able to adjust the audio up or down for those who are sensitive or hard of hearing
3. An option for subtitles will be available on video content for those hard of hearing

Physical impairment:

1. The user will be able to navigate throughout the application with only one remote for anyone with one hand or limited mobility in their hands
2. The user will be able to use the software while sitting down for users with mobility issues
3. Haptic feedback will be used to confirm a click

Intellectual impairment:

1. The app is designed for people with diverse cognitive abilities.
2. Text-to-speech will be included for users who are non-readers.

Other functionality for different communities:

1. The system will eventually be available in different languages
2. Content will not be gender specific, sexual-identity specific, or race specific
3. Universal icons will be utilized for ease of use and any icon that is unique will be explained in the tutorial.
4. A tutorial will be provided for users who do not know how to use the technology

3 . User Interface Walkthrough

This section of the document serves as a comprehensive introduction to the project's user interface (UI) design. It begins with a general overview of the various UI views incorporated in the project, accompanied by a navigation diagram (***see Figure D1***). Each mock-up screenshot is individually presented with a summary, navigation details, and explanations of buttons, text-fields, or other UI objects.

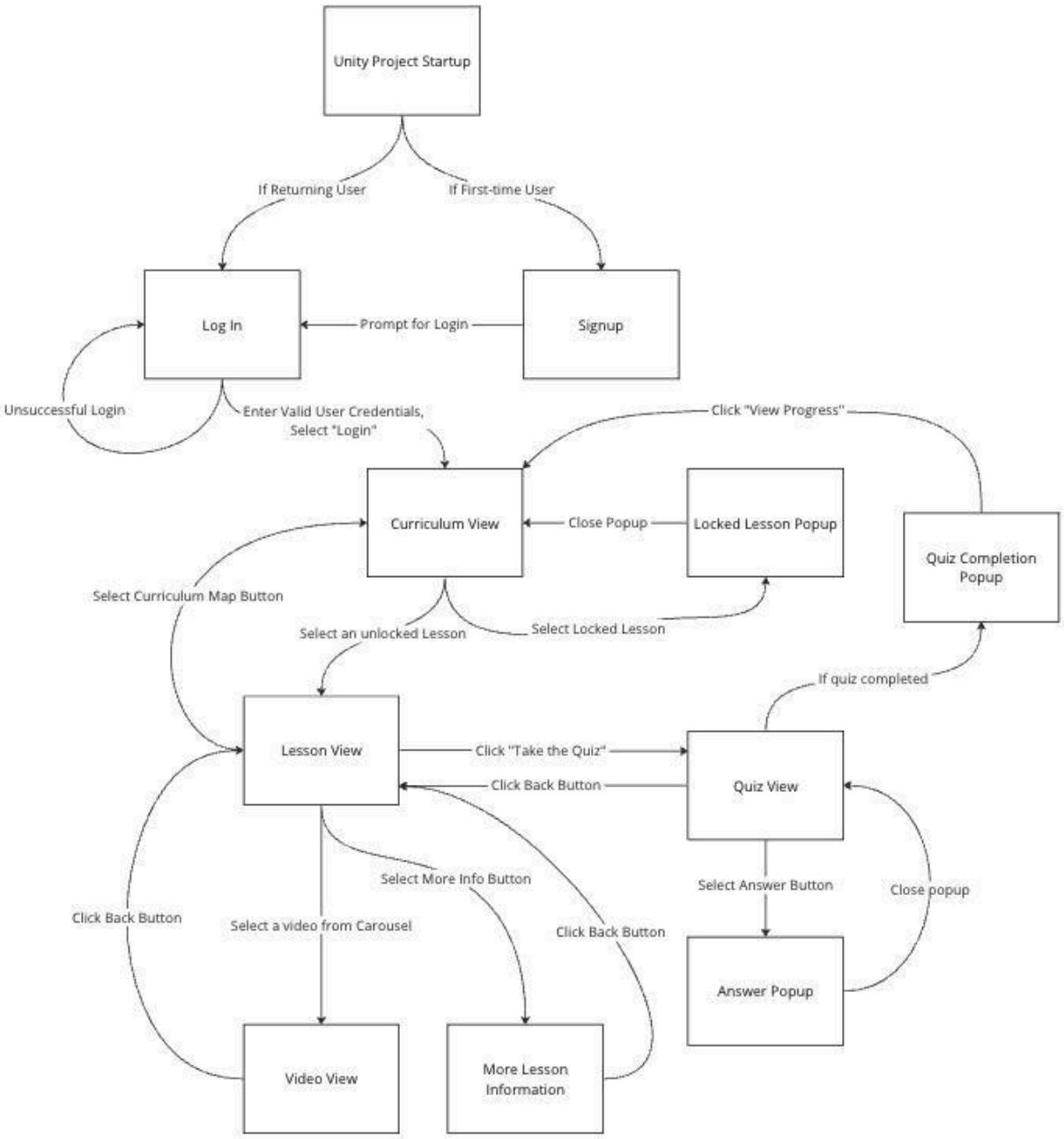


Figure D1 UI Navigation: *A detailed flow diagram depicting how a user is expected to use UI elements to navigate through the project.*

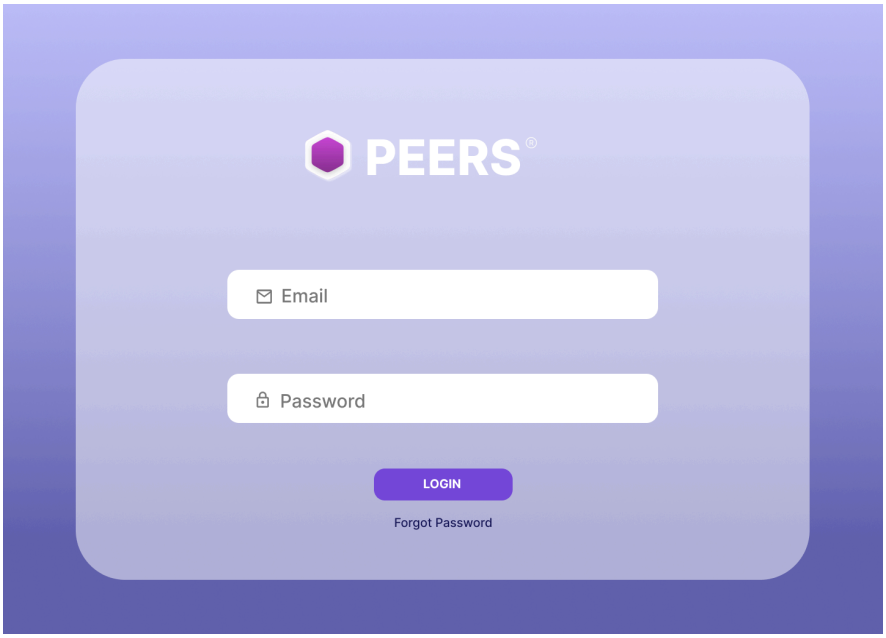


Figure D2 Login Page: *The User opens the application, they view the login page and enter their email and password.*

Summary:

The login page is the first view the user is presented with after opening the application. In order to proceed to the curriculum map, the user must first enter their account information and log in.

Navigation guide:

Once the user enters their account information and logs in by clicking the 'login' button, they will be brought to the tutorials section and then to the curriculum map, alternatively they can go straight to the curriculum map depending on if they are a new or returning user . If the user has forgotten their password, they can click the "forgot password" link and enter their email address associated with their account and instructions on how to reset their password.

Breakdown of components:

The login page contains a distinct box with the PEERS® logo at the top, two input boxes for text and a login button. The two text boxes have 'email' and 'password' written inside to indicate what information they require. There is also a "forgot password" link below the "login" button.

Database:

User login information, including emails and passwords, is stored in the user info database. Emails are stored as strings and passwords are stored as hash values.

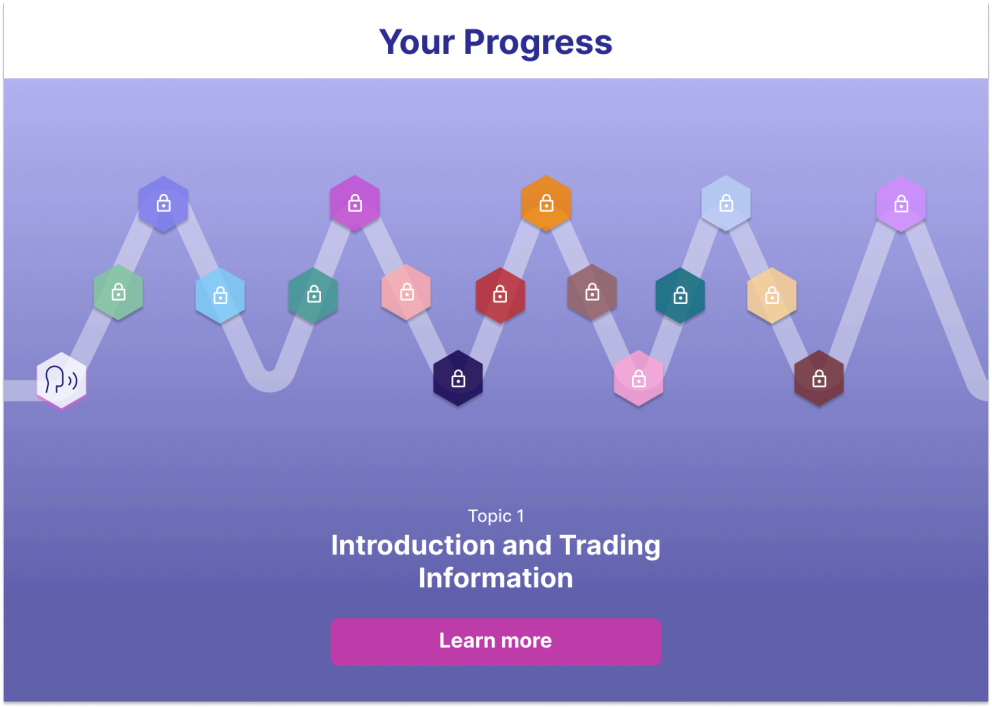


Figure D3 Curriculum map: *The home screen of the application depicting a map in the shape of a zig-zag with the user's current lesson progress.*

Summary:

The curriculum map is an extended head's up display that allows the user to look around themselves in the virtual reality environment to see the map. This will be the home screen of the application, allowing the user to access lessons and information while also telling them how much progress they have made.

Navigation guide:

Since this curriculum map functions as the home page, the user will be able to reach this screen in a multitude of ways. The user may enter their account information on the login page and click the 'login' button to reach this page or they may click the curriculum-map-shaped icon in the top left-hand corner displayed in any other screen to return. By clicking on any of the lessons visible within the screen, the user will bring up a pop-up window for that lesson, personalized to whether or not it has been unlocked yet. The user can choose which lesson they would like to interact with by clicking it, and then

by clicking on the ‘Learn More’ button at the bottom of the screen, the user can access the lesson display page for that particular lesson.

Breakdown of components:

This screen includes a zig-zag shaped trail going from left to right with several multi-colored hexagons at the peaks, troughs, and middles of every zig or zag to represent lessons. Each hexagon has either a lock symbol in the middle to indicate the lesson is still locked or a personalized icon to represent the content of the lesson in a visual. Below the map, the topic number and lesson title of the last lesson interacted with by the user is displayed. If the lesson is unlocked, there is a ‘Learn More’ button, and if the lesson is locked, there is a lock symbol and text that reads ‘Not yet unlocked’ in place of that button.

Database:

The user’s progress through the curriculum map is tracked by the data element userCurriculumProgress, which is an integer stored in the User Info Database. The integer represents the lesson number the logged in user is currently on, in other words the most recent unlocked lesson. All lessons before this number are displayed as unlocked and all lessons after are displayed as locked.

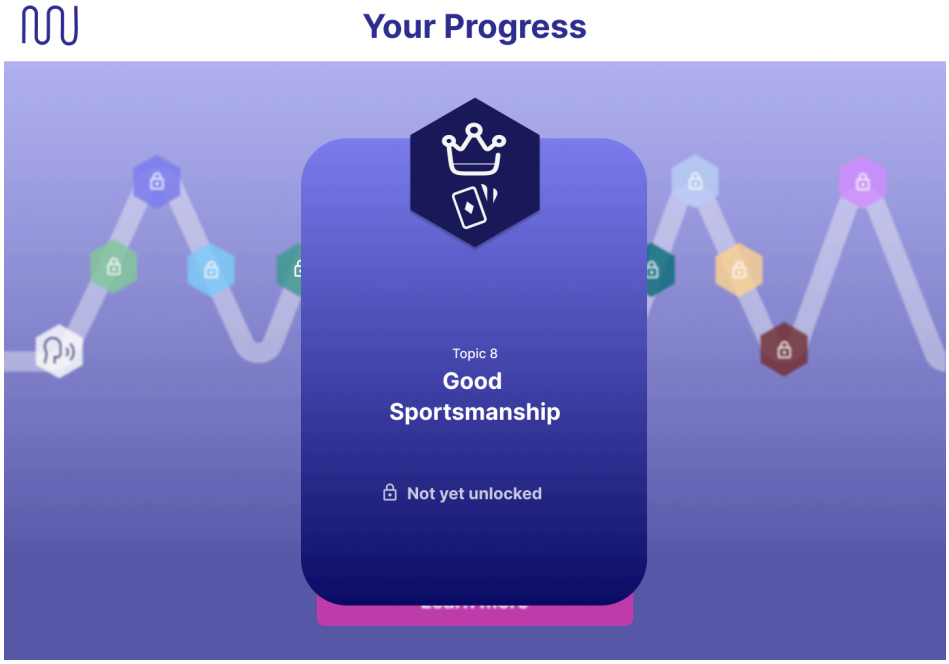


Figure D4 Locked Lesson Selection: *If the user clicks on a locked lesson this screen will pop up notifying the user the name and number of the topic along with a message that shows the lesson has not been unlocked.*

Summary:

From the curriculum map, the user can click on any lesson to be brought to the current lesson display. If this lesson is still locked the display will indicate that the lesson is 'not yet unlocked'. The user cannot enter the lesson view when it is locked, but can return to the curriculum map.

Navigation guide:

To navigate to a locked lesson the user can point the cursor at a locked lesson on the curriculum map and click. Once the locked lesson is displayed the user can return to the curriculum map by using the cursor to click on the curriculum map icon that looks like a curled zig-zag in the upper left hand corner.

Breakdown of components:

In the locked lesson display, the icon, topic number and title of the lesson are displayed as well as a lock icon and text to indicate that the lesson is locked. The user cannot continue to the lesson view. The only navigation options available from this view is to return to the curriculum map by clicking on the curriculum map icon in the upper left corner.

Database:

This UI view does not involve a query to the database. The locked/unlocked status of a lesson is established by querying the User Info Database for the userCurriculumProgress data element. This happens each time the curriculum map is rendered. The type of pop-up, locked or unlocked, depends on the current state of the curriculum map.

(UI Walkthrough continues on following page)

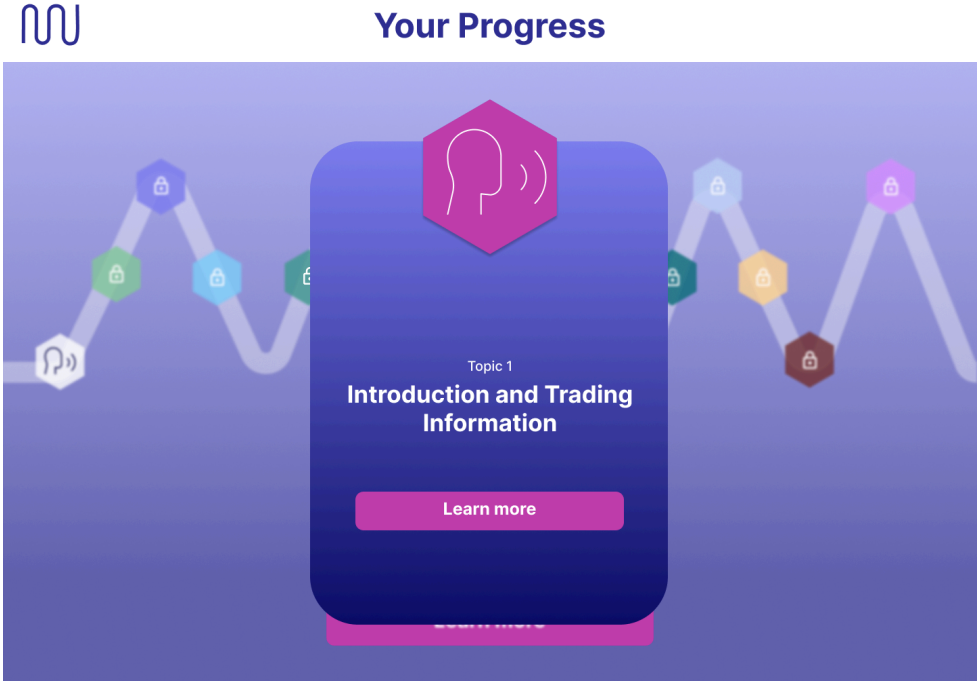


Figure D5 Current Lesson: *If the user clicks on the current lesson they are on, this screen will pop up displaying the lesson they have clicked on followed by a learn more button.*

Summary:

From the curriculum map, the user can click on any lesson to be brought to the current lesson display. If this lesson is unlocked the display will include a button reading 'Learn more' that when clicked allows the user to access the lesson content.

Navigation guide:

To navigate to an unlocked lesson display the user can point the cursor at an unlocked lesson on the curriculum map and click. Once the unlocked current lesson is displayed, the user can return to the curriculum map by using the cursor to click on the map icon in

the upper left hand corner, or continue to that lesson's lesson display view by using the cursor to click on the Learn more button.

Breakdown of components:

In the unlocked lesson display, the icon, topic number, and title of the lesson are displayed as well as a button that reads 'Learn more'. The user can navigate to the lesson view via the learn more button. Additionally, the user can navigate back to the curriculum map by clicking on the curriculum map icon in the upper left corner.

Database:

This UI view does not involve a query to the database. The locked/unlocked status of a lesson is established by querying the User Info Database for the userCurriculumProgress data element. This happens each time the curriculum map is rendered. The type of pop-up, locked or unlocked, depends on the current state of the curriculum map.

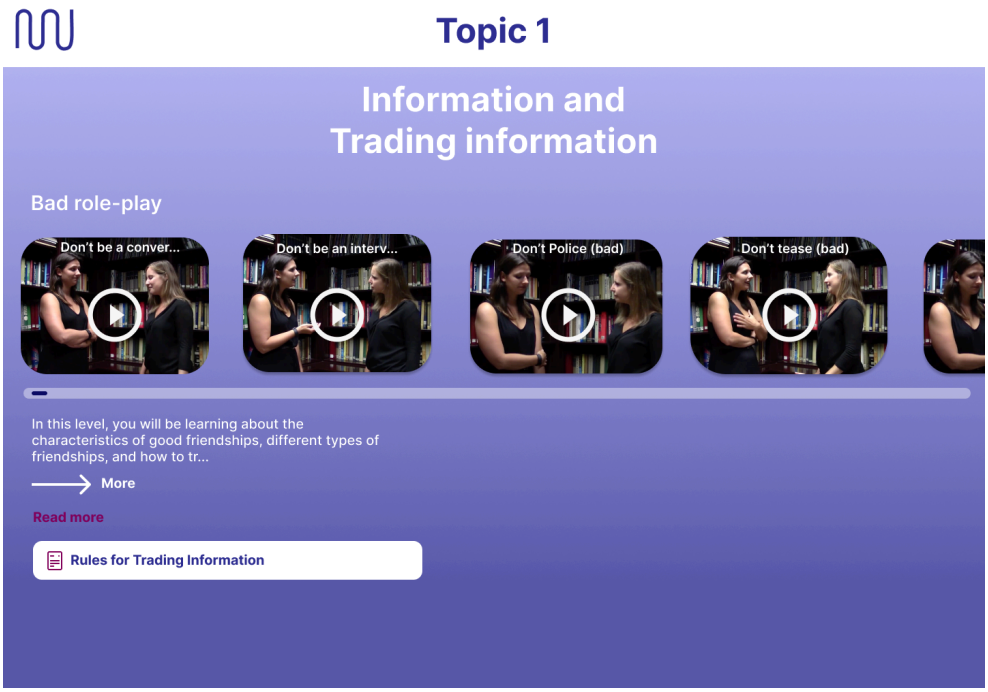


Figure D6 Lesson Display: *The lesson displays the content of the specific lesson including videos and a general description.*

Summary:

When a user clicks on a ‘Learn more’ button for a lesson, they are directed to the lesson display that originally shows bad-role play videos, a general description that can be expanded, and a button to see more content rules. The user can look down to see more lesson content.

Navigation guide:

The user can navigate to a lesson display by using the cursor to click on a ‘Learn More’ button from either a current lesson view or from the current lesson displayed on the curriculum map. The user can navigate back to the curriculum map by using the cursor to click on the curriculum map icon in the upper left corner. The user can look down to see more content like more videos and the quiz.

Breakdown of components:

On a lesson display the user is shown the title bad-role play is followed by a series of videos that showcase different examples of bad-role play in the following topic, followed by a description of what will be addressed in this lesson. The word “more” when clicked provides an extended description of the lesson. The read more section has a rules button that when clicked provides a more detailed rules and explanation of the topic. Lastly, the user can look down with the headset to see more lesson content and navigation options.

Database:

The title, labels, text information, and video thumbnails displayed in this view are all stored in the Lesson Content Database in a LessonContent object. The text information is stored in the data element lessonInfo which is formatted text. The thumbnail images for each video are contained in the data object, VideoContent. All lesson content is queried from the database each time the user enters the lesson view.

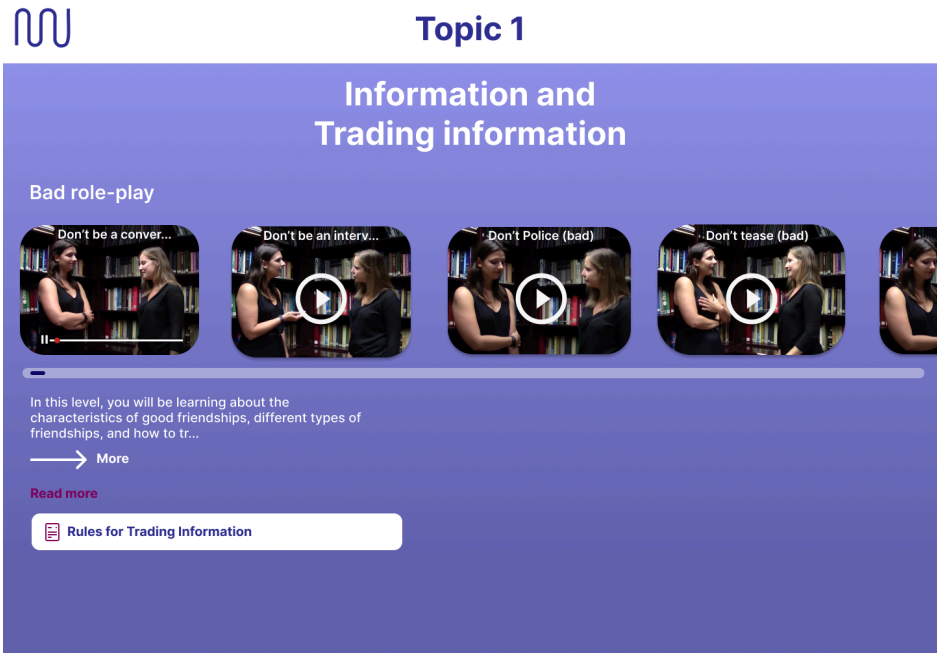


Figure D7 **Video Being Played**: *The user can click on a video in the lesson content to play the video.*

Summary:

The user has a variety of videos that they can choose to watch being displayed on the lesson view. Once the user clicks the video they want to watch, the YouTube video begins to play. The user then is provided with multiple options to enlarge the video, play/pause, and adjust the volume. The user exits back to the lesson view after the video is finished.

Navigation guide:

To navigate this view, the user must first put their cursor on the video they would like to watch, and then click the rear trigger button on the VR controller to play the video. They then have the option to pause the video by putting their cursor on the pause button in the bottom left and clicking the rear trigger button. The user can also place their cursor on the video slider, click with the rear trigger, and drag to skip to a specific time in the video. The user may also navigate their cursor to the volume slider and click the rear trigger button and drag to the desired volume.

Breakdown of components:

The video view has a pause/play button, a volume slider, a video slider, and a fullscreen button. The title of the video is displayed at the top in white text if the user is not in fullscreen mode.

Database:

When a thumbnail is clicked, the associated video file is queried from the VideoContent object in the Lesson Content Database.

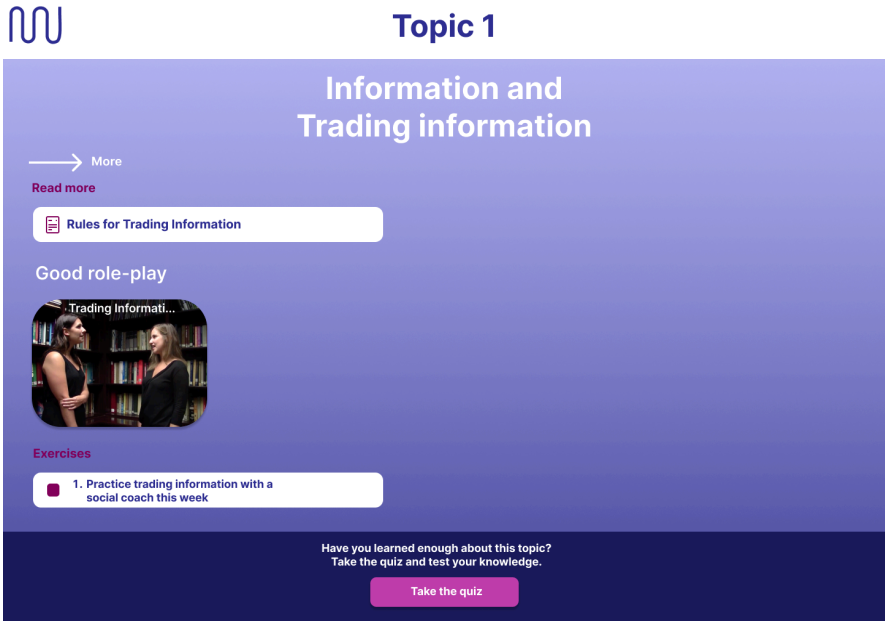


Figure D8 Additional Content: *If the user looks down they will be able to see the rest of the lesson view and lesson content including the good role-play videos, exercise and button to start quiz.*

Summary:

The exercises portion of the lesson view has multiple activities that the user can do in real life in order to improve their skills related to the current lesson. The exercises tab contains check boxes that can be filled in once the user completes the corresponding task. Once all exercises for a lesson have been completed, the “Take the Quiz” button lights up and is able to be clicked on.

Navigation guide:

In order to navigate this section of the app, the user must place their cursor on the exercises tab and click the rear trigger button, this will bring up all of the exercises for the lesson. Once in this view, the user can move their cursor over any checkbox they have completed, and again click the rear trigger button to fill in the check box. Once the user is done checking the desired boxes, they can move their cursor to the back button that is in the top left corner of the exercises tab. Once all exercises are completed the user can move their cursor to the “Take the Quiz" button and click the rear trigger button to enter the quiz view.

Breakdown of components:

There are varying buttons for the check boxes, and purple text fields to the right of them for the corresponding amount of exercises a given lesson has. There is a white“Take the Quiz” text field within a pink button located at the bottom of the page. Before completing all exercises the “Take the Quiz” text field and associated button are a darker more transparent color, and are non interactable.

Database:

The data used to create this view is already available from the first query to the Lesson Content Database which occurred when the user first navigated to the lesson view. (See Figure 6)

(UI Walkthrough continues on following page)

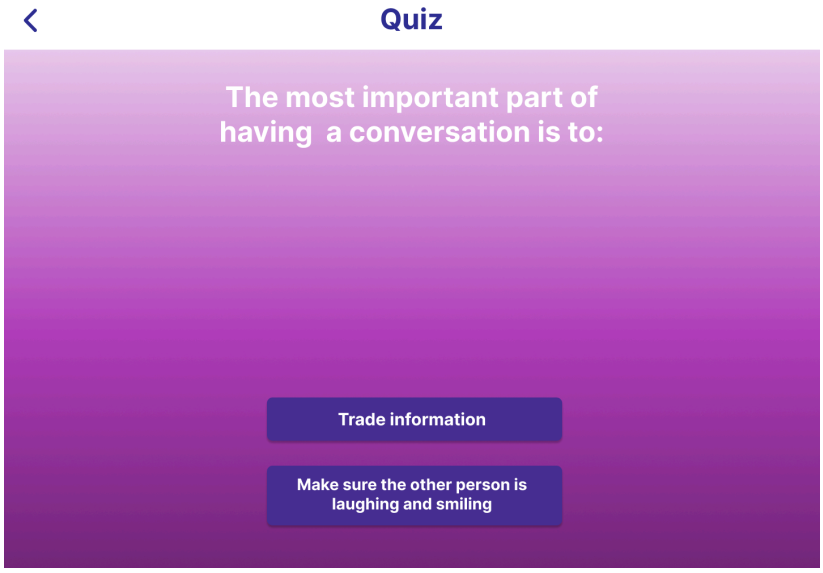


Figure D9 Quiz: *When the user clicks on take quiz they will be taken to the quiz view and they are displayed with a prompt and two choices. If the user has decided they are not ready to take this quiz the arrow on the top left will take them to the lesson view.*

Summary:

The quiz view displays a question in white text at the top of the screen, and two purple boxes that have answers in white text. The boxes can be clicked on to answer the question, and a resulting prompt will appear displaying either “correct answer” or “incorrect answer”. There is also a back arrow to navigate back outside of the quiz view.

Navigation guide:

To navigate this view, the user must aim their cursor at the button that contains the answer they would like to choose, and click down the rear trigger on the VR controller. Once the user has done this for both questions a prompt will appear for them to return to the curriculum view, which they will again press the rear trigger button to activate. If the user does not want to take the test, they may navigate their cursor to the back arrow in the top left corner of the view and press the rear trigger button to return to the lesson view.

Breakdown of components:

There are two buttons on the bottom of the view that can be clicked to answer the question, and one button in the top left that can be clicked to return the user to the lesson view. There is one text field at the top of the screen reading “Quiz”, one text field below

that reading the question, and two different text fields within the two answer buttons that display each potential answer.

Database:

The quiz prompts, answers, and results are all stored in a QuizContent object in the Lesson Content Database. Upon successful completion, the userCurriculumProgress integer is incremented in the User Info Database to reflect the user's progress.

4 . Data Validation

The following offers a detailed breakdown of each data item that can be entered into the project. Table D1 provides specific information about data items, including data types, size limits, boundary cases, and allowable formats. Notably, this project requires very few data items from the user, aside from those necessary for login and authentication.

Table D1: Data Validation

Data item	**Data type**	**Limits**	**Allowable format(s)**
username	string	64 bytes	Must be an email Cannot exceed 64 characters
password	string/hash	128 bits / 16 bytes	Must be at least 8 characters long Must include ● At least 1 number ● At least 1 uppercase letter ● At least one special character Cannot exceed 16 characters

name	string	35 bytes	Cannot exceed 35 characters Only include letters

Appendix D - CIR

Code Inspection Report

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

March 23, 2024

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno,

Jacob Michaud, Maha Fazli

Virtual Video Modeling on the Social Skills of Adults with Autism

Code Inspection Report

Table of Contents

1. Introduction

This is a project for Dr. Sarah Howorth on virtual video modeling of the social skills of adults with autism. Dr. Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS® curriculum, a formal process for teaching social skills, was written by Dr. Elizabeth Laugeson. Our client's project proposal outlined a request to make the curriculum available in a virtual reality environment.

The JamTech team consists of five students currently enrolled in a two-semester (Fall 2023 - Spring 2024) computer science capstone course at the University of Maine. Each member of the team has volunteered for a specific team role which allows us to delegate assignments to everyone on the team.

1.1 Purpose of This Document

The document is a record of all aspects of JamTech's code review process. Each section of this document corresponds to a specific component of the review process. Section 1 includes JamTech's adopted coding conventions and a checklist of code defects which violate these conventions. Section 2 contains JamTech's personal impressions of the code review process as well as records of the individual inspection meetings. Section 3 is primarily a detailed list of every module that was critically reviewed. Lastly, section 4 contains a table in which every defect was enumerated, categorized, and documented. The intended audience of this document is our client, Dr. Sarah Howorth, and our capstone professor Dr. Laura Gurney.

1.2 References

- *C# at Google Style Guide*. styleguide. (n.d.). https://google.github.io/styleguide/csharp-style.html
- PEERS®. (2021). PEERS® (version 1.1.0) [Mobile app]. Apple Store OR Google Play. https://play.google.com/store/apps/details?id=com.peersclinic.peers&hl=en_US&gl=US
- PEERS (2023) UCLA PEERS® Clinic, Semel Institute for Neuroscience and Human Behavior. https://www.semel.ucla.edu/peers
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). GitHub repository, https://github.com/VoloVita/SeniorCapstone/tree/main/Deliverables
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). System Design Document (SDD), version 1.0

- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). Systems Requirement Specification (SRS), version 2.0

1.3 Coding and Commenting Conventions

For coding and commenting conventions, we have decided to follow the guidelines set by Google's style guide for C# code. This guide was developed internally by Google and has since become Google's default guide. These guidelines include naming conventions, file naming conventions, whitespace rules, and overall organizational conventions.

Link to Source: https://google.github.io/styleguide/csharp-style.html

The naming convention for classes, methods, enumerations, public fields, public properties, and namespaces is PascalCase, where the first letter of each word is capitalized. We will be using camelCase for local variables and parameters, where the first letter of each word, except the first word, is capitalized. _camelCase with an underscore at the beginning will be used for private, protected, internal, and protected internal fields and properties. For all of these previously mentioned naming conventions, a 'word' is defined as anything written without internal spacing.

Names of all interfaces will begin with 'I'. Files, filenames and directory names will use PascalCase, and when possible, file names should match the name of the main class within the file. Our files should generally be limited to one core class per file. Whitespace will limit us to one statement per line, and we will intent with tabs, which are five spaces wide, except when in yaml files. Additionally, all trailing whitespace is trimmed to keep the code concise. The overall organization of the guide states that namespace 'using' declarations always go at the top of the file. Class members also have a specific order which they must follow, beginning with nested classes, enums, delegates, and events; static, const, and read-only fields; fields and properties; constructors and finalizers; and finally methods.

1.4 Defect Checklist

In this section, we've provided an organized table of all defects found in our code through the inspection process.

Table 1. Defect Checklist: *A list of common defects found in code.*

#	Defect	Category	Coding convention	Produces Errors?
1	improper usage of Update()	Logic Error	Update is called every frame, this is not suitable for all operations and can cause error	maybe
2	Improper usage of Start()	Logic Error	Start is called upon program startup and is not suitable for all operations and can cause errors.	maybe
3	Improper _camelCase	Naming Convention	Applies to private, protected internal and protected internal fields and properties	no
4	Improper PascalCase	Naming Conventions	Applies to: classes, methods, enumerations, public fields, public properties, namespaces	n/a
5	Improper camelCase	Naming Conventions	Applies to local variables and parameters	no
6	Non-Descriptive Naming	Naming Conventions	Chosen names should be useful and descriptive	no
7	Improper Acronyms	Naming Conventions	All character in acronyms should all be capitalized	no
8	Improper File names	Naming Conventions	Filenames and directory names are PascalCase	no
9	File Structure	Organizational	In general, prefer one core class per file.	no
10	Lack of documentation	Organizational	Every non-standard method should have documentation	no
11	Class Variables not defined at the top of class definition	Organizational	variables should be defined at the top of	no
12	Public and Private Variables Mixed	Organizational	Variables should remain private unless they are needed to be public, and that	no

			declaration should remain consistent	
13	Incorrect public/private class definition	Security Oversight	Only certain needed information should be public whereas almost all other variables should	no
14	Personal Identifiable Information not obfuscated	Security Oversight	All sensitive data should not be kept in plain text	no
15	Improper whitespace	Whitespace Conventions	One statement per line, indentation with tab (5 spaces)	no
16	Trailing whitespace	Whitespace Conventions	Avoid excessively spaced out formatting.	no

2. Code Inspection Process

The goal of a code inspection is to review the code of the project, making sure everything is clean, well organized, functional, and bug free. This process focuses on locating bugs and defects within the code so that they can be fixed, and analyzing the internal issues within the code. Our code development is currently still in progress. Within these code inspections, we analyzed our scripts for convention issues, errors, and security oversights.

2.1 Description

Our team began every meeting with a friendly and casual atmosphere as we noted the date, time, and location. After everyone was settled and ready to focus, we started with a brief overview of the agenda for our code inspection meeting. This agenda began by reviewing the defect checklist we constructed. We also identified the modules that would be inspected during that meeting, and which sections of code we would be talking about in detail.

After the overview, we would assign roles and responsibilities for the meetings. These roles included presenters and notetakers. One person in the team was a presenter for each code inspection review, and one person was a notetaker. Everyone else listened and offered input and feedback throughout the course of the meeting.

Code inspection began with the presenters introducing their code, walking the rest of the team through it and ensuring everyone understood the content and functionality. This

presentation of code would be followed by a Q&A and group feedback. The process of recording action items would follow, where we discussed as a team what the defects were and how to address them. The notetaker for the meeting would note all defects discussed, and as a group we would decide how to avoid each of them going forward. At the end of the code inspection review, we would summarize all modules analyzed throughout the meeting and make sure there were no outstanding questions.

2.2 Impressions of the Process

Overall, our team has a positive impression of the code review process. It's a great method for making sure the whole team has seen every aspect of the project, and it's good practice to see code written by other members of the team with enough understanding to be able to pick it apart. The code review process encourages better documentation of code and it ensures that our code is more consistent across the project, and that the code itself contains fewer errors. This makes our code structure and formatting more consistent. However, the process was boring as there were some files that yielded little-to-no defects or errors after review. The process of checking for every possible defect felt tedious. This does not, of course, negate the necessity of critical analysis.

Additionally, resolving a defect often led to other errors throughout the file and the project. For instance, if we decide as a team to change the name of a specific variable to meet conventional standards, we then have to go through every bit of code to update all references to that variable, and then re-run our testing to ensure that nothing else was broken.

If there was one thing we could do differently as a team, we agree that we would have set convention standards before beginning the coding process specifically for the reason mentioned previously. Defining naming conventions before programming reduces the possibility of having to go back and make changes later. We believe this would have saved us a significant amount of time. In terms of our units, our best modular unit would be Jsonconfig.cs. This unit is small and simple, and therefore yielded few errors. On the other hand, our worst unit was by far lesson1test.cs due to its large size and complexity. We found the most errors in this unit, and it's the most likely to continue having errors in the future as development continues.

2.3 Inspection Meetings

Our group has had two code inspection meetings for the purpose of reviewing and discussing our code in depth. Most of our code is C# scripts, but a lot of development involves standard components of the Unity Editor, which means that not everything can be reviewed as code.

We spent our meetings reviewing our scripts and discussing all possible defects and errors within them. All members were present for the code inspection review and all members participated in reviewing the presented scripts.

Table 2. Inspection Meetings Held: *A brief description of each code inspection held, the location, time, the participants and their roles as well as the code units reviewed.*

Date	Location	Start	End	Performed By & Role	Participant/s	Particular code unit covered
2/28/24	Library Study Room - in person	5:30pm	6:30pm	Jacob - presenter Maha - notetaker	Tristan Cilley, Jacob Michaud, Maha Fazli, Allison Lupien, Nick Sarno	Player.cs Jsonconfig.cs PlayerData.cs
3/4/24	Library Study Room - in person	5:00pm	6:30pm	Jacob - presenter Nick - notetaker	Tristan Cilley, Jacob Michaud, Maha Fazli, Allison Lupien, Nick Sarno	Lesson1test.cs LessonData.cs SaveSystem.cs

3. Modules Inspected

This section contains a detailed description of each module that was evaluated during our code review process. Every description contains an SDD reference which explains whether the class is new or a modified version of a class outlined in the SDD.

Module Name: Player.cs
Description: A custom script for saving and loading player data.
Context: This script defines the Player class. It contains two public methods which are used for saving or loading the state of the player.
SDD Comparison: This class most closely represents the PlayerObject class defined in figure 2.2 of our SDD. In addition to storing the player's position, this class will be expanded to save and load additional detail from PlayerData.cs

Module Name: Jsonconfig.cs
Description: This module handles the loading of lesson data from a JSON file.
Context: Jsonconfig.cs loads the lesson data from a JSON file located at the path within the project's asset directory.
SDD Comparison: In the SDD we planned to use a database to store lesson content. We are using JSON files in the place of having the Lesson Content Database. The video content is

now stored in our Assets folder and the JSON file contains the file paths for each lesson's videos.

Module Name: PlayerData.cs
Description: This is a custom script that holds player data such as the lesson the player is on and the player's position in the scene.
Context: The PlayerData class holds the players state information and the Player class can access this information as needed.
SDD Comparison: This class contains information from the CurriculumView class seen in figure 2.2 from the SDD. Mainly the lesson variable is synonymous with the userCurriculumProgress variable. It has the additional variable for the player's position. It has the potential to hold additional player data as development progresses that may include data from the Lesson and Quiz classes from the SDD.

Module Name: lesson1test.cs
Description: This is a custom script that allows for dynamically loading and allocating data into the lesson panel within the scene.
Context: The lesson1test class loads the JSON file containing the lesson data based on an integer value passed into the main function of the class: lessonLoad(). Supporting functions include one for loading video sprites into the scene from JSON: LoadSpriteFromFile(), and for changing the video displayed in the video view: ChangeVideo().
SDD Comparison: This class is not specifically mentioned in the SDD, but does contain the functionality mentioned in the LessonView scene. This class takes the methods, (selectlesson(), selectvideo(), and getlessonprogress()) mentioned from the lessonview scene and combines them into a fully functional all-in-one implementation.

Module Name: LessonData.cs
Description: Two classes detailing specific aspects of each lesson.
Context: This module assists with importing lesson data from JSON files into the framework of the lesson view.
SDD Comparison: This class was originally listed as LessonInfo in our SDD. This script deals with linking the lesson data to the actual lessons, but whereas in our SDD we planned to pull the information from a database, LessonData pulls from JSON files containing the lesson information.

Module Name: SaveSystem.cs
Description: A class which provides methods for saving and loading the player's state
Context: This class contains two functions; SavePlayer() writes all the player's data to a file after serializing it, PlayerData() deserializes the contents of the save file and loads it into the current session.

SDD Comparison: Neither this class nor its component functions were outlined in the SDD. This is a novel class for our project. Figure 2.2 of our SDD includes a class for players which contains a field for the player's position in the scene. Our current implementation saves the player's position but its functionality has been expanded to include other aspects of the current state.

4. Defects

This section contains a table of all defects discovered in the code review process. Every defect has a general and specific categorization as well as references to its exact location in the code.

(Table on following page)

Table 3. Found Defects: *A list of all defects found during the code review meetings including the module or file they appear in, the kind of defect it is, what the defect is, the component it applies to, and the line of code it is found.*

Module Name	Defect Category	Defect Specific	Component Name:	Line #
Player.cs	Naming Conventions	Non-Descriptive Naming	variable: lesson	**7**
Player.cs	Organizational	Lack of documentation	function: SavePlayer()	17
Player.cs	Organizational	Lack of documentation	function: LoadPlayer()	23
Jsonconfig.cs	Logic Error	Improper usage of Start()	function: Start()	12-14
Jsonconfig.cs	Organizational	Lack of documentation	function: LoadLessonData()	16
PlayerData.cs	Naming Conventions	Non-Descriptive Naming	variable: lesson	9
PlayerData.cs	Naming Conventions	Non-Descriptive Naming	variable: position	10
PlayerData.cs	Organizational	Lack of documentation	function: PlayerData()	13-20
lesson1test.cs	Naming Conventions	improper naming PascalCase	variable: testobject variable: lessonPannelcontent variable: header variable: description variable: non_example_content variable: example_content variable: moreinfo public class: lessonload(int i)	11 12 13 14 15 16 18 48

Module Name	Defect Category	Defect Specific	Component Name	Line #
lesson1test.cs	Naming Conventions	Non-Descriptive Naming	variable: videop variable: butt variable: badVid	18 98 and 128 96
lesson1test.cs	Naming Conventions	improper camelCase	variable: videoreset variable: videoreset2 variable: badthumb variable: goodthumb variable: goodvideocount variable: badvideocount	53 54 59 60 61 62
lesson1test.cs	Naming Conventions	improper _camelCase	variable: lessonData	17
lesson1test.cs	Naming Conventions	improper filenames	file: lesson1test.cs	9
lesson1test.cs	Organizational	Public and Private Variables Mixed	all public variables	
LessonData.cs	Naming Conventions	improper camelCase	variable: non_example_videos variable: more_info variable: example_videos variable: quiz_url	12 14 15 17
LessonData.cs	Organizational	File Structure	public class: lessonData public class: Lesson	4 10

Module Name	Defect Category	Defect Specific	Component Name	Line #
LessonData.cs	Organizational	Public and Private Variables Mixed	variable: lesson1 variable: lesson2 variable: title variable: non_example_videos variable: description variable: more_info variable: example_videos variable: exercises variable: quiz_url	5 6 11 12 13 14 15 16 17
LessonData.cs	Naming Conventions	Improper PascalCase	public class: lessonData	4
SaveSystem.cs	Organizational	Lack of Documentation	public class: SavePlayer()	**7**
SaveSystem.cs	Organizational	Lack of Documentation	public class: PlayerData()	19
SaveSystem.cs	Whitespace Conventions	Trailing Whitespace	public class: PlayerData()	29

Appendix E - AM

Administrator Manual

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

April 15, 2024

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno,

Jacob Michaud, Maha Fazli

Virtual Video Modeling on the Social Skills of Adults with Autism

Administrator Manual

Table of Contents

Page

1. Introduction

This is a project for Dr. Sarah Howorth on virtual video modeling of the social skills of adults with autism. Dr. Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS® curriculum, a formal process for teaching social skills, was written by Dr. Elizabeth Laugeson. Our client's project proposal outlined a request to make the curriculum available in a virtual reality environment.

The JamTech team consists of five students currently enrolled in a two-semester (Fall 2023 - Spring 2024) computer science capstone course at the University of Maine. Each member of the team has volunteered for a specific team role which allows us to delegate assignments to everyone on the team.

1.1 Purpose of This Document

The intended audience is Dr. Howorth and other researchers. The purpose of this document is to inform administrators on how to run and maintain the application in order to use this application as a research tool. It gives an overview of the hardware and software requirements, how to install the application, common tasks that need to be performed, where to find user support, and how to troubleshoot common issues.

References

- PEERS®. (2021). PEERS® (version 1.1.0) [Mobile app]. Apple Store OR Google Play. https://play.google.com/store/apps/details?id=com.peersclinic.peers&hl=en_US&gl=US
- PEERS (2023) UCLA PEERS® Clinic, Semel Institute for Neuroscience and Human Behavior. https://www.semel.ucla.edu/peers
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). GitHub repository,https://github.com/VoloVita/PeersVR_Capstone/tree/main/Documentation/Deliverables
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). System Design Document (SDD), version 1.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). Systems Requirement Specification (SRS), version 2.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). User Guide (UG), in progress

2. System Overview

This is an overview of the system for the system administrator. The following sections summarize daily operations and maintenance, and lists the technical requirements of operation.

2.1 Background

The system administrator is responsible for acquiring the source code, installing the application, performing routine administrational tasks, and providing user support. There are specific guides on each of the administrator's responsibilities including forking the PeersVR_Capstone Github repository, building the .apk from the source code, and loading the .apk onto an android based headset. The PEERS® VR program does not require any routine maintenance. Additionally, administrative tasks may also include expanding or adding content to lessons. These tasks are explained in more detail in section 3: Administrative Procedures.

2.2 Hardware and Software Requirements

In order to run and update the application, there are specific hardware and software requirements that need to be met depending on the mode of execution.

Running the Application on a VR Headset:

- The application was designed for, and tested on, the Oculus Quest 1 headset, but it should be compatible with any version of the Oculus Quest or newer.

Running the Application in Unity w/o Headset:

- Windows Operating System (recommended)
- Unity Editor version *2021.3.34f1* (required)
- Dedicated GPU (recommended)
- Unity XR Simulator Plugin (recommended)
- Unity XR Interactive ToolKit (recommended)

Running the Application in Unity w/ Headset via Quest Link:

- Windows Operating System (recommended)
- Unity Editor version *2021.3.34f1* (required)
- Dedicated GPU (required)
- USB-C cable (recommended)

3. Administrative Procedures

The administrator procedures will assist users in both installing and maintaining the system, while also providing access to user support as required.

3.1 Installation

The following section explains the process by which a new administrator can acquire the source code, build the project, and load it onto a VR headset. GitHub has detailed documentation for the use of their platform, we recommend that anyone unfamiliar with GitHub should familiarize themselves with basic operations before continuing.

Acquiring the Source Code:

The source code for the PEERS® VR program can be acquired from the GitHub repository. It is recommended that the new administrator forks this repository to ensure that they possess an identical copy of the project without being dependent upon the availability of the original repository. Once the project has been forked, the new administrator should clone the repository. This will create a copy of the project's source code in the administrator's local environment. The entirety of the source code will be contained inside a folder called "*PeersVR_Capstone*".

Opening the Unity Project:

In order to view the project inside the Unity Development Platform, the new administrator must have a Unity account. Sign-up for a Unity account.

Additionally, the administrator must also have Unity Hub installed to open and manage Unity Projects. PEERS® VR was developed using the Unity editor version *2021.3.34f*. This unity installation is required in order to open the project from Unity Hub. Unity editor version *2021.3.34f* can be installed through Unity Hub, or from the Unity website. To view the project inside Unity, open Unity Hub and add a project. This action should prompt the administrator to specify where the project files are located. As mentioned previously, the project will be contained inside a folder called "*PeersVR_Capstone*".

To reduce the size of the GitHub repository, the project folder only contains essential files. This means that Unity will have to create additional folders such as Library, Build, and User Settings. This process only occurs once and will happen automatically the first time the project is opened from Unity Hub. Due to the size of additional folders needed, it may take a few minutes.

Acquiring the PEERS® Video Content:

The PEERS® VR program contains media content in the form of example roleplay videos and thumbnail images. Since this content is private property, it is not included in the public GitHub repository and must be downloaded separately. JamTech will provide this content to the new administrator via a private folder on Google Drive. This folder must be downloaded and placed inside the JamTech_assets folder at: *PeersVR_Capstone/Assets/JamTech_Assets*

Building an .apk from source code

Following this YouTube tutorial, the administrator will be able to download the appropriate android SDK and JPK addons for Unity in order to be able to build the executable .apk(android/VR) file. After downloading and installing these, if they continue following the instructions of the tutorial they will set up and optimize the project for the Oculus device via build settings and project settings in Unity. Lastly, when these settings are completed and they are ready to create an executable, they press the build button and set a local location on their computer which it will be stored in.

Loading the .apk onto headset:

In order to be able to load the .apk file onto a headset standalone, you will need to have access to a developer account for Unity, which essentially enables you to put non licenced applications that you have created or others onto the device. The account you use does not have to have special permissions other than being a developer account. Once you have access to and are logged into the developer account on your headset, download the SideQuest application. If you have created and verified the unity account, this app will guide you to setting up the device and downloading the app onto it. Once you download the app onto the oculus, go into apps→unknown sources→the name of the app you built. This will run the application on the headset.

3.2 Routine Tasks

For this project, there aren't many maintenance tasks that need to be done. If for any reason there is a bug during runtime it is recommended to restart the application.

3.3 Periodic Administration

The PEERS® VR program does not require regular maintenance, but it is possible to expand the content of the program to include more of the PEERS® curriculum.

Adding Additional Lesson Content:

As of the conclusion of the Capstone course, the first five lessons of the PEERS® curriculum are fully functional. It is possible to expand the program to include more lessons, but it will require additional labor. Each additional lesson will require the creation of a new folder (named: topic_<lesson#>) inside the PeersVideosContent directory. Inside the topic folder two more folders will be created, one for video files (mp4) and another for thumbnail images (jpg). The .meta files are created automatically by Unity once the project is opened.

Figure 1. Lesson Content: *Each lesson requires a folder of necessary files, including thumbnails and videos.*

The program stores the information for all lessons inside a JSON file at: *PeersVR_Capstone/Assets/JamTech_Assets/JSON/lesson.json.* This file will need to be expanded to include the text information of the new lesson (title, description, etc) as well as the relative file paths to the new video and thumbnail images. It is absolutely vital that the file paths match the names of the files

The program stores the information for all quizzes inside a JSON file at: *PeersVR_Capstone/Assets/JamTech_Assets/JSON/quiz.json*. This quiz file will also need to be expanded to include the text information of quiz questions associated with each new lesson.

Note: When adding new thumbnail content to a unity project, it will be necessary to set the texture type of the file to ‘Sprite (2D and UI)’ inside the inspector panel (as seen in Figure 2 below)

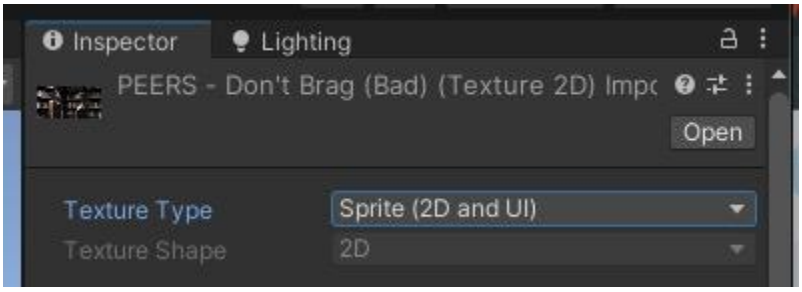


Figure 2. 2D Sprite: *Images should be turned into 2D Sprites by changing the Texture Type.*

Additionally, new ReadMore panel content will need to be added to the main scene of the project (as seen in Figure 3 below).

Figure 3. Lesson Objects in Scene Hierarchy: *Each lesson needs an object under the ReadMore panel, as seen in the scene hierarchy view.*

3.4 User Support

To acquire user support for this document, please refer to these specified deliverables on the application GitHub:

- **JamTech_AM :** which is the Administrator Manual, the manual contains all the information needed for installation, troubleshooting, etc..
- **JamTech_UG :** which is the User Guide, maybe a quick description of pdf, this document’s purpose will be for any guidance that is intended for the team or developers.

For further details and assistance, kindly reach out to us via the team email address: JamTech397@gmail.com

4. Troubleshooting

This section will cover troubleshooting methods and responses to error messages, failures, or known bugs within our program, including what they are and how to deal with them if or when they occur.

4.1 Dealing with Error Messages and Failures

To handle errors and failures while expanding, the administrator will need to perform manual testing. Manual testing can be performed by entering play mode in Unity, interacting with the scene and the elements under development. The developer can then change elements in play mode to see realtime effects. It is important to note that changes made in play mode will not be saved. To save changes, move back into edit mode, make the necessary changes and save. The new changes can be manually tested by entering back into play mode.

If an error of failures occurs while using the current application, the user should restart the application.

4.2 Known Bugs and Limitations

1. **Summary:** When scrolling the content of the lesson view vertically, it is possible for the cursor to get stuck in the horizontally-scrolling content containing example and non-example videos which prevents the user from scrolling vertically.
   Location: Lesson View Panel
   Solution: We can add colored backgrounds to the scrollable areas to help users distinguish between them.

2. **Summary:** The quiz view does not provide feedback for correctness of responses like it should.
   Location: Quiz View Panel
   Solution: We can add the visual pop up which informs the user if their quiz response was correct or not.

3. **Summary:** The curriculum view needs a second step for selecting a lesson to display which lesson is being selected.
   Location: Curriculum View Panel
   Solution: Selecting a lesson from the curriculum panel should be one action. Transitioning from the curriculum view to the lesson view should be a separate action.

4. **Summary:** When building the executable, the lesson content does not load to the lesson view when selected.
   Location: Discrepancy between script elements and inspector details.
   Solution: It is possible to assign object references in the inspector panel by executing a script. This will ensure that the program builds successfully.

5. **Summary:** Only the first 5 lessons are implemented so far; the rest will redirect the user to the first lesson.
   Location: lesson.json, quiz.json, and PeersVideosContent
   Solution: For more details, see section 3.3: Periodic Administration.

6. **Summary:** No UI element programmed to exit the application.
   Location: N/A
   Solution: The program either needs to be stopped in the Unity Editor, or closed in the VR headset.

Appendix F - UG

User Guide

for

Virtual Video Modeling on the Social Skills of Adults with Autism

Version 1.0 approved

For Sarah K. Howorth

University of Maine

April 27, 2024

Prepared by JamTech:

Tristan Cilley, Allison Lupien, Nick Sarno,

Jacob Michaud, Maha Fazli

Virtual Video Modeling of the Social Skills of Adults with Autism
User Guide

Table Of Contents

1. Introduction

This is a project for Dr. Sarah Howorth on virtual video modeling of the social skills of adults with autism. Dr. Howorth is the director of the PEERS® Lab at UMaine and our primary client for this capstone project. The PEERS® curriculum, a formal process for teaching social skills, was written by Dr. Elizabeth Laugeson. Our client's project proposal outlined a request to make the curriculum available in a virtual reality environment. The JamTech team consists of five students enrolled in a two-semester computer science capstone course at the University of Maine during the Fall 2023 - Spring 2024 semesters. Each team member has volunteered for a specific team role, which allows for assignments to be delegated to everyone on the team. The intended audience of this document is our client, Dr. Sarah Howorth, and our capstone professor, Dr. Laura Gurney, as well as any future users of the PEERS® VR application.

1.1 Purpose of Document

This document provides the user with a comprehensive guide to using our application. It includes explanations of all relevant features, operations, errors, and instructions pertaining to PEERS® VR.

1.2 References

- PEERS®. (2021). PEERS® (version 1.1.0) [Mobile app]. Apple Store OR Google Play. https://play.google.com/store/apps/details?id=com.peersclinic.peers&hl=en_US&gl=US
- PEERS (2023) UCLA PEERS® Clinic, Semel Institute for Neuroscience and Human Behavior. https://www.semel.ucla.edu/peers
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). GitHub repository, https://github.com/VoloVita/PeersVR_Capstone/tree/main/Documentation/Deliverables
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). System Design Document (SDD), version 1.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). Systems Requirement Specification (SRS), version 2.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). User Interface Design Document (UIDD), version 1.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). Code Inspection Report (CIR), version 1.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2024). Administrators Manual (AM), version 1.0
- Tristan Cilley, Allison Lupien, Nick Sarno, Jacob Michaud, Maha Fazli. (2023). GitHub repository, https://github.com/VoloVita/PeersVR_Capstone/tree/main/Documentation

1.3 Overview

This system aims to teach social skills to adults with autism in a more engaging and immersive environment than the existing PEERS® mobile application. This User Guide aids users in understanding and interacting with the system. This document is intended to guide both new users and laymen as well as serve as an official user guide for the project.

1.4 How to Use this Document

The intended use of this document is to provide the user with a comprehensive guide on how to use our application. This document includes explanations of all relevant features, operations, errors, and instructions pertaining to PEERS® VR. Section 1 of this document is mainly an overview of the project, its related documents, and how it can be used. Section 2 is the system overview, which discusses the background of the project as well as the basic operations of the system. Section 3 provides a detailed explanation on how the user interacts with the system. Section 4 will outline any error messages that might be encountered, as well as give a diagnosis and suggest recovery procedures.

All the sections in this document complement each other as they help enhance the reader's understanding of the application, from a general overview of the project to specific usage instructions on how to operate the system. By the end of this document, the user will have all the necessary knowledge to use this application.

1.5 Related Documents

All related documents can be found in the GitHub repository in the deliverables directory.

Table 1. Document: *This table contains the name and author of each document created so far during the development of this project, as well as the date it was completed and the date it is issued or given to the client.*

Num	Title	Author	Date	Issued
1	System Requirements Specification	JamTech	11/27/2023	5/3/2024
2	Systems Design Document	JamTech	11/15/2023	5/3/2024
3	User Interface Design Document	JamTech	12/4/2023	5/3/2024
4	Critical Design Review Document	JamTech	12/15/23	5/3/2024
5	Code Inspection Report	JamTech	3/23/2023	5/3/2024
6	Administrator Manual	JamTech	4/15/2023	5/3/2024

2. System Overview

Anyone can use this application, but PEERS® VR's target demographic and user base are adults with autism and educators. The system was designed to be run by an administrator in a research-based setting with a participant acting as the user. The application can be run from the Unity editor or a VR headset. To run in Unity, the program's source code must be downloaded from Github and opened as a Unity Project. In this scenario, an administrator or third party viewer can monitor the user through the connected computer. This application must be loaded onto the headset as an executable file to use only a VR headset. Though it is possible to expand the content in this application, no regular updates or maintenance are needed.

To use this application, the user progresses through multiple educational lessons teaching and demonstrating content pertaining to social skills through both written word and roleplay example videos. After viewing the content of the lesson, the user can take a quiz to proceed with the remainder of the lessons.

The application is designed to work with VR hand-held remotes, and is not compatible with newer VR headsets that employ hand-recognition software. Our program utilises either the right or left remotes' trigger to select any buttons or drag through scroll boxes, and the remotes' joystick as an additional scrolling mechanism. Outputs for our application include the VR headset's visual experience and the audio output of the roleplay videos.

Anyone can use our application with a VR headset, but it's recommended that users sit down to avoid dizziness or nausea and to take breaks to avoid eye strain.

3. Instructions

In this section, we've included instructions for every step when using our application, including the individual operations, descriptions of each, warnings or errors if there are any, and a detailed procedure for the user to follow for each operation. These operational instructions assume a system administrator has already completely set up the program to run in Unity. Any operational instructions that reference headset mechanics or adjustments refer to the Meta Quest 2 headset specifically.

Operation: Running the application from Unity w/ VR Headset
Description: Assuming the system administrator has properly set up and configured the program and a compatible VR headset is connected via QuestLink, running the application is as simple as playing the scene in the Unity Editor.
Cautions/Warnings: Please take stock of your surroundings and sit comfortably before running the application.
Probably Errors & Causes: If Unity produces warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Precheck 1: The PEERS® VR source code has been successfully downloaded and configured by the system administrator

Precheck 2: The Unity Project has been launched from Unity Hub using a compatible editor version

Precheck 3: Meta Quest Link application has been launched.

Precheck 4: VR headset is connected to computer via USB-C

Running the Program: At the top of the Unity Editor there are controls for running, stopping, and pausing the scene (**see figure 1**). Select the play arrow to play the project.

Figure 1. Play and Pause: run and pause the scene.

Operation: Opening the application from executable on headset
Description: Using the Oculus headset to locate and launch the Peers VR application.
Cautions/Warnings: The application may be located in the "unknown applications" section of the apps on the device because it has not been published.
Probably Errors & Causes: N/A
Procedures:

Locating the App: Assuming the application has already been put on the headset through developer mode, it should now be located in the app's tab of the headset. If you still cannot

see the application, it is likely in the “unknown apps” section which can be accessed by selecting the filter at the top and then selecting “unknown applications”.

Launching the App: After locating the app, launching it is as simple as clicking on the icon for the application and then selecting the “launch app” option.

Closing the App: To close the application, press the meta button on the right controller to open the menu and select the “quit” option.

Operation: Using VR safely
Description: When using a VR Headset, it is important to be aware of your surroundings and sit in a stable chair.
Cautions/Warnings: Check for pets, other people, furniture, or other inanimate or animate objects that may obstruct your experience.
Probably Errors & Causes: N/A
Procedures:

Check Surroundings: If you are seated in a chair that can spin or swivel, please fully extend your arms away from you and spin slowly to ensure that you will not collide with your environment during the operation.

Operation: Movement within the VR scene
Description: Using the Oculus headset controls to adjust the user's physical location within the scene.
Cautions/Warnings: Make sure to be seated and within the Oculus Guardian boundaries
Probably Errors & Causes: It is possible to stand inside or behind the floating view panels. To correct this, either move backward in the scene or restart the application.
Procedures:
(Meta Quest 2 Safety Manual: https://www.meta.com/quest/safety-center/quest-2/#manuals)

Moving: Use the analog stick on the left controller to move in any direction,

Turning: Use the analog stick on the right controller to turn by flicking it left or right, and use the Oculus headset to look around within the scene.

Operation: Adjusting VR headset volume
Description: Using the Oculus headset controls to adjust the auditory output from the headset.
Cautions/Warnings: N/A
Probably Errors & Causes: If the headset volume controls are not working as expected, please consult the headset’s safety manual for possible solutions.
Procedures:
(Meta Quest 2 Safety Manual: https://www.meta.com/quest/safety-center/quest-2/#manuals)

Watch the tutorial as your headset loads or consult the headset’s manual for specific instructions. Each headset will have volume controls at a different location with different

methods of functionality, so the procedure for adjusting the volume will be unique to the user's headset.

Operation: Adjusting VR headset focal point
Description: Using the Oculus headset controls to adjust the visual focal point of the VR lens.
Cautions/Warnings: N/A
Probably Errors & Causes: If the headset focal adjustment controls are not working as expected, please consult the headset's safety manual for possible solutions.
Procedures:
(Meta Quest 2 Safety Manual: https://www.meta.com/quest/safety-center/quest-2/#manuals)

Watch the tutorial as your headset loads, or consult the headset's manual for specific instructions. Each headset will have focal adjustment controls at a different location with different functionality methods, so the procedure for adjusting the lens will be unique to the user's headset.

Operation: Selecting a lesson within the curriculum view
Description: Using the controller to select a lesson from the curriculum view.
Cautions/Warnings: N/A
Probably Errors & Causes: If Unity produces warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Horizontal scrolling within curriculum view: Use the front trigger on the left or right controller to select the curriculum view background or the scroll bar at the bottom of the panel and move your controller back and forth to scroll through the curriculum icons.

Selecting Lesson: Use the front trigger on the left or right controller to select one of the lesson icons on the panel. Then use the front trigger to select the "Learn More" button.

Operation: Navigating the Lesson view
Description: Using the controller to interact with the Lesson view.
Cautions/Warnings: N/A
Probably Errors & Causes: If Unity produces warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Selecting the curriculum view back button: The curriculum view button is located in the upper right-hand corner of the lesson panel.

Figure 2. Curriculum Button: The curriculum navigates back to the curriculum view

Use the trigger on the front of the right or left-hand controller to select the button to navigate back to the curriculum view.

Scrolling vertically in the lesson view: There are a few different ways to scroll vertically through the lesson view:

a. Point a controller at the scrollbar on the right side of the panel, and then click and hold the corresponding front trigger button while moving your hand up and down.
b. Point a controller at the background of the panel, then click and hold the corresponding front trigger button while moving your hand up and down.
c. Push the analog stick of the right hand forward to scroll up and back to scroll down.

Scrolling through the videos horizontally: There are a few ways to scroll horizontally through the video content:

a. Point a controller at the horizontal scrollbar under the video thumbnails, then click and hold the corresponding front trigger button while moving your hand left and right.
b. Point a controller at the background between the video thumbnails then click and hold the corresponding front trigger button while moving your hand left and right.
c. Push the analog stick of the right-hand left to scroll left and right to scroll right.

Operation: Completing lesson exercises
Description: Within the lesson view, the user may engage in exercises that pertain to the current lesson.
Cautions/Warnings:
Probably Errors & Causes: If Unity produces warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Locate the purple checkbox and prompts towards the bottom of the lesson panel. Aim the right controller at the area with these prompts and once hovered over, press the rear trigger button on the right controller. Once the user has read and or completed the exercises, locate the “X” in the top right corner of the pop up panel. Aim the right controller at the “X” and once hovering over, press the rear right trigger button to return to the lesson view.

Operation: Viewing more lesson info.
Description: Within the lesson view the user may view more information describing the current lesson.
Cautions/Warnings:N/A
Probably Errors & Causes: If Unity produces warnings or errors in the editor console window, please inform the system administrator.

Procedures:

More info or read more: When on the lesson view for a particular lesson, locate the "more info" button, which is located just below the non-example role-play video content, or the "read more" button, which is located just below the "more info" button. Aim the right controller at the button, and once hovering over it, press the rear trigger button on the right controller. You should now see a panel containing more information about the lesson. When finished reading, locate the "X" in the top right corner of the popup screen, hover over it, and press the rear trigger button on the right controller.

Operation: Navigating the video view
Description: After selecting the desired video, learn how to watch the video, pause and play, seek through to a specific timestamp, and return to the lesson view.
Cautions/Warnings: Audio warning: if the video is too loud or too quiet, you may want to adjust the headset volume, or if running from unity on a computer, adjusting the computer volume.
Probably Errors & Causes: If Unity is producing warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Back button: In order to return to the lesson view, locate the back arrow button in the top left corner of the video view panel. Aim your right controller at the arrow and once hovered over, press the rear right trigger button on the right controller to return to the lesson view.

Multimedia controls: Upon entering the video view, the video selected should begin to play automatically. If the user wishes to pause the video, aim the right controller anywhere within the video that is playing. Once hovering over the video press the rear right trigger button on the right controller to pause the video. In order to resume the video, repeat the same process.

If the user wishes to skip to a specific time stamp in the video there is a scroll bar below the video that is being displayed. Aim the right controller at the scroll bar and press and hold the rear right trigger button on the right controller, then drag the scrolling point left or right to the desired timestamp location by aiming the controller. When the desired location has been selected, release the rear right trigger button to jump to that timestamp.

Operation: Navigating the quiz view
Description: Taking a lesson quiz
Cautions/Warnings: Exiting out of the quiz view before answering both questions will not save your progress or unlock the next lesson.
Probably Errors & Causes: If Unity is producing warnings or errors in the editor console window, please inform the system administrator.
Procedures:

Back button: In the top left corner of the panel, a back arrow is displayed that can be aimed at with the right controller. Once you are hovering over the arrow, press the right rear trigger on the right controller to return to the lesson view.

Quiz question and answers: At the top of the Quiz view, there will be a question for the user to answer. After reading the question, there are two possible answers at the bottom of the quiz view to choose from. To select an answer, aim the right controller at the desired answer and press the rear right trigger button on the right controller. A prompt will appear telling the user whether they got the correct answer. To continue to the next question aim at the continue button at the bottom of the prompt and again press the rear right trigger button.

4. Reference Section

This section will cover error messages the user may receive while using the application and the recovery procedures for resolving them. Some of these error messages can only be resolved by editing the source code opening the project in the Unity Editor. The system administrator should be contacted in both cases to fix the problem.

4.1 Error Messages and Recovery Procedures

Error: Thumbnails not set to 2D or Sprite

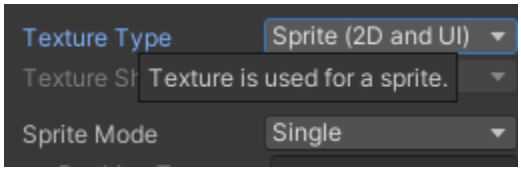


Figure 3. Thumbnail Texture Type Source: Selecting the Sprite (2D and UI) option.

Recovery Action: At the bottom of the Unity window is an area for viewing the project's files. In that view, navigate to the folder that contains the thumbnail images. Select one or many image(s) and then open the inspector panel. At the top of the inspector panel, open the drop-down menu for Texture Type and select the 'Sprite (2D and UI)' option. Finally, scroll down to the bottom of the inspector window and select the Apply button.

Error: File path is Null
Recovery Action: The lesson text content, quiz text content, videos, and video thumbnails are all accessed via file paths, which are stored in their corresponding JSON files. If the file paths are not correctly written, then when the program tries to load the thumbnail image, a NullReferenceError will be produced (Fig 4). The JSON file should contain relative file paths like the one below
Example of Thumbnail path:
"Assets/JamTech_Assets/PeersVideosContent/topic_1/thumbnails/PEERS - Don't be an interviewer (bad).jpeg"

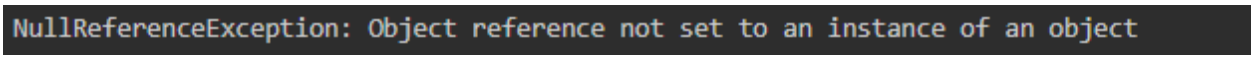


Figure 4. NullReferenceException: The error message displayed when there is a missing link between objects in the scene.