CEN 6940 Project Proposal

Project Title: CyberSphere

Project Category: Innovative Product

Team Name: VR Cyber Educators

Submitted By: Thang, Taiwo, and Zoe'

Executive summary:

Our problem results from the archaic practice of teaching methods for cybersecurity-related topics. Many existing programs rely largely on abstract concepts, creating a less user-friendly experience by limiting hands-on practice and real-world problem-solving opportunities. Our innovative product produces three cybersecurity sections: cybersecurity techniques & threats, threat analysis, incident response, and interactive training. Each section teaches users about a security-related topic where they can gain hands-on experience and interact with real-life problems.

We evaluated our product by testing multiple features and functionality. The Oculus Quest was constantly used to test the outcome of each section we controlled; based on the results received from testing, we would make improvements and change assets, scripts, etc. The main issues faced when producing the project were technical issues with the Oculus Quest. Each team member had difficulties connecting the Oculus Quest to our devices, limiting us only to the VR labs on the university's premises. The contributions of each team member are:

- Thang: This section is Incident Response and Iterative Training. The user/student will be introduced to IR (Incident Response) scenarios and the Escape room offers an interactive way of simulating a real-world environment and teaches how to handle cybersecurity challenges.

- Zoe’: This section is Threat Analysis. It offers an introduction to malware (Malicious Software) and in conjunction with digital forensics, digital evidence (i.e. an infected image) can be analyzed by the user/student to identify the tactics used by cybercriminals and help create an effective counter. The user/student will also learn about steganography by analyzing the infected image.
- Taiwo: This section is Cybersecurity Techniques and Threats. Phishing and Encryption are opposing sides of a coin. The user/student will learn about ways in which criminals use social engineering to steal data while also learning about techniques to protect data from such threats.

Introduction:

Cybersecurity education is often hindered by outdated, and theory-heavy teaching techniques that more often fail to offer practical experience. Students lack the chance to replicate dynamic cyberattacks or apply defensive techniques in a controlled and realistic environment. This space in education eventually leads to unprepared individuals going into the workforce, increasing an organization's vulnerability to cyber threats.

The continuous evolution of cyber threats requires skilled workforces that are capable of providing effective responses. Without practical experience, even the smartest individual would find difficulty adapting to the dynamic nature of real-world incidents, leaving businesses at risk. Tackling this issue is vital to making sure sensitive assets and information are secured. Our team is motivated by the need to aid change in cybersecurity education. As students and professionals with a drive for this field, we understand the potential of virtual reality to change learning by offering immersive, hands-on training thus bridging the gap between theory and practicality.

We opted for an iterative design approach, focusing on user engagement and modular learning. By leveraging VR technology and its ability to offer diverse and interactive scenarios, we developed a comprehensive training tool that provides an immersive hands-on learning experience for users. Through this, we can combine with traditional teaching methods thus enhancing the overall experience, as well as offer a path towards designing additional modules for interactive learning tailored to specific skill sets and levels.

Our product CyberSphere, offers three modules:

- Cybersecurity Techniques & Threats: Covers phishing and cryptography through the encryption of data.
- Threat Analysis: Introduces malware analysis and digital forensics.
- Incident Response: Offers a real-world simulation of a breach and guides users towards addressing it.

By undertaking this project, we expected to enhance users' level of readiness towards addressing real-world challenges, improve their knowledge retention through its interactive and engaging nature, and finally, offer a scalable virtual reality tool for practical use in education. Here are the general goals we laid out for this project:

- Provision of an immersive learning experience.
- Allow users to apply their knowledge in a realistic scenario.
- Creation of modular tools that are adaptable to the user's needs.
- Ensure it can be used for diverse settings i.e. education and professional.
- Prepare users for real-world cybersecurity roles.

The end goal of this project is to strengthen users with the confidence and skills to tackle complex threats in cybersecurity. With the main audience our project targets being students

(college level), teachers/professors, and industry professionals (Entry-level IT roles), organizations will also benefit as they will be receiving a more prepared workforce in turn enhancing overall security and lowering risks.

This project aims to benefit stakeholders by equipping users with the skills demanded by employers, allowing teachers/professors to implement practical tools in their teaching, and aiding organizations to improve their security defense through well-trained professionals. By addressing the gaps in traditional cybersecurity education, CyberSphere offers to the growing field, showcasing how VR can be implemented to enhance learning and prepare users for real-world challenges.

Problem definition:

The specific problem our product aims to address is the outdated teaching methods used in cybersecurity that rely heavily on theory for a field where the practical application is vital resulting in less-than-confident students and professionals thus increasing the risk of security breaches. The increasing amount of complicated cyberattacks lays more emphasis on the need for skilled people with practical experience making this problem a cross between education, technology, and workforce preparedness increasing its level of significance and complexity.

What is currently known is the fact that conventional teaching methods are not sufficient enough and practical training does improve skills gained. What is currently left to be known is how far virtual reality can bridge the gap between theory and practice as the field itself is still evolving. The key points for our proposed solution lie in providing the following:

- Immersive hands-on scenarios and environments where users are not limited to what they can learn.

- A modular-based and flexible design that allows for scalability.
- A product that is accessible to diverse users and groups.

Based on the above key points, CyberSphere, a VR-based learning platform is offered through its combination of immersion, interactivity, and multi-topic coverage that ensures users are better prepared.

In our Incident Response section, users are exposed to realistic cyberattack scenarios where they can practice addressing breaches and replicating real-life cyber incidents that require immediate action to be taken. This will help users build upon their critical thinking and problem-solving skills needed for handling incidents.

As cybersecurity professionals must constantly protect data from ever-changing threats, a major issue is the underdevelopment of practical skills in mitigating these threats. In the Cybersecurity Techniques and Threats section, we address this by focusing on the practical application of data protection methods such as encryption functions. Users will also learn about techniques used by cybercriminals such as phishing and tips to spot likely phishing attacks thus providing an understanding of technical details they may not consider while learning from a theoretical point of view.

A vital problem in cybersecurity education is the lack of realistic tools for threat analysis and malware identification. Students commonly learn about malicious software in an abstract sense, never being offered a chance to see how such threats are born in a real-world scenario. In the Threat Analysis section of CyberSphere, modules are designed where users analyze malware such as infected images, and use digital forensics tools to uncover hidden data. By offering students the chance to engage in malware analysis and digital forensics in an interactive sense, they develop the necessary skills to locate, respond to, and prevent such threats.

Background:

In the field of cybersecurity, incident response is among the most vital areas professionals must be adept in. It involves effectively addressing cybersecurity breaches, making sure the damage is minimized and the incident is contained. However, conventional methods of teaching incident response primarily focus on theoretical knowledge which leaves students unprepared for the fast-paced and unpredictable nature of cyber incidents. By incorporating virtual reality scenarios, students can practice their decision-making skills and fine-tune their ability to respond under pressure in real-world conditions.

The Incident Response and Iterative Training module of CyberSphere enables users to learn how to handle breaches in a simulated environment, where they can make mistakes, learn from them, and adjust their responses over time. This approach will help students gain the confidence and experience needed to deal with cybersecurity incidents in the real world.

Cybersecurity threats are constantly changing, and one of the most pressing issues in cybersecurity currently is ensuring that professionals are equipped to handle a variety of attacks, from phishing scams to malware infections. Conventional training methods tend to focus on theory, but they fail to address the practical aspects of preventing and defending against these attacks. The Cybersecurity Techniques and Threats module in CyberSphere addresses this gap by immersing students in real-world cybersecurity situations where they can directly learn and engage with the methods used by cybercriminals.

Through interactive simulations, users will not only learn about threats like phishing but will also be able to apply their knowledge by practicing how to defend against such attacks. For example, students will have the chance to test encryption techniques. This hands-on experience

in combating cyber threats provides students with the necessary skills needed for a career in cybersecurity.

Another significant issue in cybersecurity education is the need for students to understand how cyber threats such as malware function, and how to analyze them using tools like digital forensics. Malware analysis and forensics are essential skills that are often ignored in traditional cybersecurity training, where students may only learn about these topics from a conceptual viewpoint. The Threat Analysis section of CyberSphere allows students to experience firsthand how cybercriminals employ malware and hidden data to carry out attacks.

By creating a VR module where students can interact with infected images and perform malware analysis, they can learn how to detect, dissect, and neutralize threats. This section not only enhances students' technical abilities but also prepares them for the challenges they will face in real-world cybersecurity roles.

Methodology:

The development of CuberSphere followed an iterative process, ensuring the design and practical functionality:

- Investigation of Requirement: Identification of challenges and user needs in cybersecurity education.
- Design of Concept: Creation of initial scenarios and module concepts.
- Development of Prototype: building and testing a basic virtual reality environment.
- User Feedback: Using user input for further improvements.
- Final Optimization: Adjusting the product and improving its effectiveness and usability.

This methodology allowed for continuous improvement, making sure that our product CyberSphere met both technical and educational requirements while addressing the challenges of VR development.

Innovative Product Description:

Overview of capabilities: Our innovative model produces three topics of security for users to learn through. By incorporating Phishing & Encryption, Malware & forensics, and Incident response, users can receive hands-on experience through multiple topics of security. The issue we aim to address is the disconnect between conventional cybersecurity education and the practical/real-world experience students require to be more proficient in the field. Current methods require students to constantly learn in classroom settings where they can gain theoretical knowledge but lack on-the-job experience. Cybersecurity threats are constantly changing and evolving and students need to gain hands-on experience to learn in such a field.

CyberSphere consists of :

- Phishing & Encryption (Taiwo): Assist users with identifying phishing attacks. The users will interact with an interactive puzzle-based scenario and be introduced to encryption keys in hidden messages.
- Malware & Forensics (Zoe'): Users will be introduced to steganography where they will use commands learned through the lesson. Once they use the correct command, they will be introduced to malware. During this phase, they will identify the type of malware they are interacting with.

- Incident Response (Thang): Allows users to engage with a realistic incident response scenario and escape room simulation. The user will encounter a server breach where they must navigate through traps to disconnect power while experiencing flashing lights.

Competing Products:

The article “CyberVR - An Interactive Learning Experience in Virtual Reality for Cybersecurity Related Issues” implemented an interactive VR game for users to learn about the cyber field through multiple mini-games provided through the virtual experience. Their goal is to create a tool to educate and improve users' knowledge of cybersecurity. They also decided to use an Oculus Rift and Unreal engine to create their game while we are using an Oculus Quest headset and Unity. Our model will allow users to interact with multiple categories and topics. For example, we have three games that allow users to fully understand the topics instead of mini-games. Allowing the user to have multiple topics to learn about through each game allows them to obtain a better understanding of security.

The article “Exploring Cybersecurity Education and Training Techniques: A Comprehensive Review of Traditional, Virtual Reality, and Augmented Reality Approaches.” produces a summary of different educational techniques in cybersecurity while comparing them to traditional methods with innovative approaches. A VR learning model mentioned in the article implemented current cyberattacks and countermeasures to teach users about protection against attacks. Even though our model includes multiple cyberattack instances, we also incorporated additional cybersecurity topics such as encryption and digital forensics. This allows the user to understand additional tools they might encounter in the cybersecurity field.

New about our product: Our interactive model will introduce a new learning tool for students and workers where they will receive hands-on experience in cybersecurity which can be difficult without work experience. The model provides users with multiple topics that they can learn about. For example, if they would like to improve their skills on commands used in digital forensics, then they can select the Malware and forensics topic, to better identify potential phishing emails as well as how to decrypt sensitive information, they can select the Phishing and Encryption module, to experience a simulation of a real-world incident response and steps to navigate through it, they can cover the IR module.

Features & capabilities:

The model will allow users to gain both theoretical knowledge and hands-on experience while actively learning about cybersecurity. Our simulation-based learning method introduces users to a day-to-day working life where they can use their experience against real-life problems. By enhancing cybersecurity education, we can increase the number of users who can gain hands-on experience, especially since threats are changing daily. Users will use Oculus Quest controllers to traverse the environment to solve puzzles, interact with items, and learn different topics.

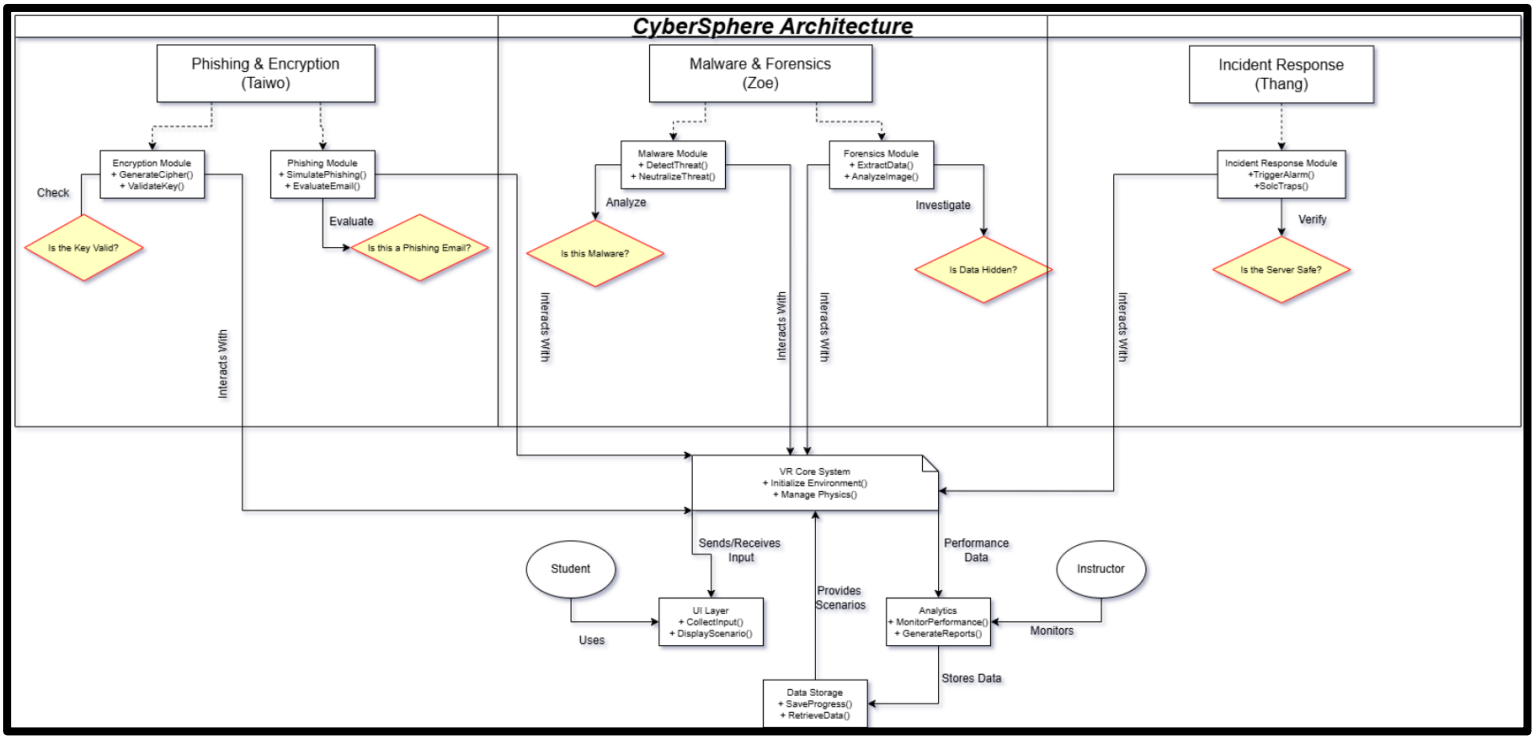
Some key features of our product include:

- An Immersive VR Training Environment: A simulated VR Training environment where students dive in to experience real-world scenarios such as server rooms, forensic labs, and office settings. This will help combat the lack of hands-on training by simulating an environment where students practice real-world cybersecurity day-to-day tasks.
- Independent Module Design: Separated training modules that cover various cybersecurity topics:

    - Phishing & Encryption Module: Teaches about identifying phishing emails and offers safe tactics. Use of a cipher decryption in the module introduces cryptography providing the chance to learn about encryption.
    - Malware & Digital Forensics: Mimics detecting and neutralizing malware (malicious software) while also guiding students in analyzing hidden data.
    - Incident Response: Simulates an office crisis scenario to improve critical thinking and response strategy.
- Scenario-Based Learning: Students are introduced to practical challenges such as analyzing emails, message decryption, breach response, etc. These scenarios serve to develop and improve critical thinking and decision making thus addressing the issue of the gap present between theoretical knowledge and real-world application.
- Gamified Learning: Offering a sense of achievement through the incorporation of achievements and rewards encourages friendly competition as well as improving motivation and knowledge retention.

Architecture and Design:

UML Diagram Link: [CyberSphere Architecture]



Software Architecture Overview: The architecture is a layered system that consists of several interconnected components:

- VR Core System: This is the backbone layer that creates and manages the immersive environment, controlling the rendering of 3D scenes, user movement, and physics interaction. We made use of the Unity game engine due to its user-friendly features that make adapting to game development less strenuous.
    - Architectural Link: Based on the input received from the UI, the VR core system performs the intended task such as loading the requested scene from the Data Storage. Performance data is also received and sent from/to the Analysis module.

- Modules: Specialized training segments consisting of unique cybersecurity concepts such as Phishing to evaluate email authenticity, Encryption to test cipher validation, Malware to detect and neutralize threats, Digital Forensics to analyze hidden data, and Incident Response to guide users in handling server breaches. To achieve this, C# scripting was incorporated to achieve a coherent flow in the scenario and challenges provided.
    - Architectural Link: Direct interaction with the VR Core System for loading the various scenarios and physics, Offers feedback to the UI, and Sends results to the Analysis module.
- User Interface (UI): Covers the displaying of scenarios and instructions, accepting user Input, and providing reports based on said input. To achieve this, we made use of Unity's Canvas System and EventSystem for managing the users' input and button interactions,
    - Architectural Link: Send the user input to the target modules, direct interaction with the VR Core System.
- Analysis: Monitors the users' performance, providing hints, and a general progress report. Through scripting, we send the users' performance to be stored for later use.
    - Architectural Link: Stores progress data in the data storage, and receives users' performance from modules via the VR Core System.
- Data Storage: Maintains scenario configurations.
    - Architectural Link: Offers the stored scenarios to the VR Core System, and saves the users' performance data received from the Analysis module.

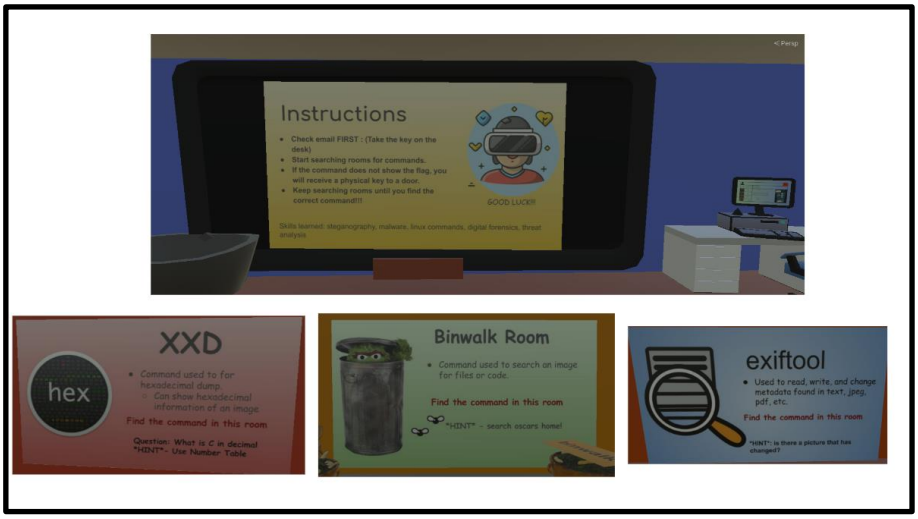
This architecture provides a vast, immersive, and interchangeable solution that effectively addresses the issue in cybersecurity education, offering practical training in a simulated yet realistic environment that ensures students are more prepared to tackle real-world challenges.

Cybersecurity Techniques & Threats:

- Lead Contributor: Taiwo
- Responsibilities:
    - Designed the phishing scenario to evaluate the users' ability to detect email authenticity.
    - Scripted realistic email templates with hidden threats and safe indicators using unity
    - Developed encryption scenario along with interactive encryption-decryption tool to test users’ ability to validate ciphers
    - Incorporated scenario progression logic for user advancement.
- Architectural Link:
    - Direct Interaction with the VR Core System for generating emails and tracking users' responses.

- Sends data to the Analytics module for feedback generation.

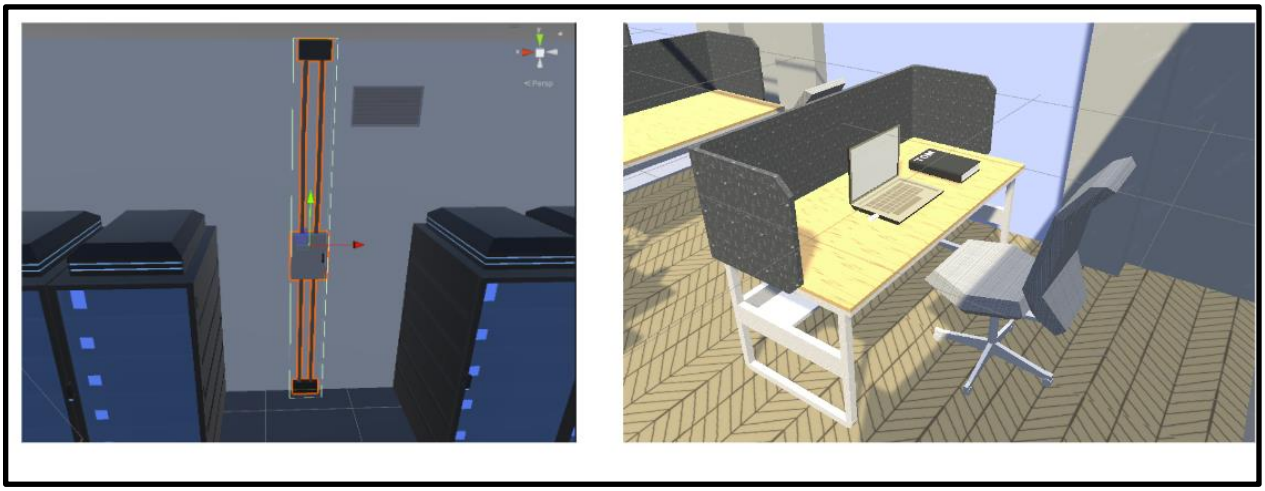
Threat Analysis:



- Lead Contributor: Zoe’
- Responsibilities:
    - Created a malware detection training module to identify and neutralize threats.
    - Implemented guided hints to assist users when detecting hidden threats.
    - Scripted digital forensics scenarios to analyze and extract hidden data.
    - Integrated Unity asset management tools to create authentic digital forensic and malware scenes.
- Architectural Link:
    - Sends data to the UI for displaying instructions as well as results.

- Shares results with the analytics module for detailed reporting and user performance analysis.

Incident Response & Interactive Training:



- Lead Contributor: Thang
- Responsibilities:
  - Developed an incident response training simulation to guide users through handling server breaches.
  - Incorporated Unity's scripting tool to simulate real-time threat mitigation.
- Architectural Link:
  - Shared user inputs with the analytics module for feedback.
  - Accessed scenario configurations from Data Storage.

Design Rationale:

Justificatory Knowledge: To design the product, our group utilized the overall experience and knowledge base that is present within each group member. As students with an interest in the field of cybersecurity, the group had a baseline knowledge of cybersecurity based on the curriculum taught at the University of North Florida. For the design of the virtual reality simulator, our group member, Thang, had previously completed projects in his undergraduate course involving game design within virtual reality. We utilized his experience with the technology to help navigate and utilize the technology to produce our product. Along with that, is his experience as a security analyst, which brings in a real-world perspective on cybersecurity, differing from the academic perspective taught in school. Taiwo provides his AutoCAD skills in designing potential models for the team to use in the product. Zoe' brings knowledge from previous experiences through her certificate studies in security-related topics.

Principles of Implementation:

Our current and final design of the project was based on several factors. The first factor that the group considered was how we could produce a product that would require a higher level of engagement from the end user. We wanted to consider that the majority of end users who would be playing this game would be new to the field of cybersecurity, and the potential of how some information could not be engaging to them. As such we decided to utilize virtual reality as our platform for the challenges. Another factor that we have taken into account is the flexibility that the virtual reality environment provides to us when adapting to the guidelines presented to us in the class. By utilizing virtual reality as our platform, we are given a certain level of creativity and freedom for designing our product while still following the guidelines that have been set for the various deliverables. The final factor that we took into account for choosing to design our project

is the flexibility in the application at all levels of audience (i.e., university, workplace, etc.). With the design of our product, it can be easily applied to people from a wide variety of backgrounds, making it easier to utilize and teach to a wider audience.

Iterative Process:

The design process of CyberSphere followed an approach that involved multiple phases of planning, testing, and refining. This allowed the team to better come up with alternative solutions and approaches we could take toward reaching our ideal goal/design that would also meet the project requirements.

The steps taken in the process were as follows:

- Requirement Investigation: Determining the main challenges in cybersecurity education was the first task our team underwent. Objectives for the VR system which included real-world scenario generation, immersion, etc. were also laid out as well as how we would tackle them.
- Concept Design: Initial sketches and conceptual scenarios were developed for what the VR game entailed.
- Prototype Creation: Based on the conceptual design, our team built a basic layout of the VR scenarios to gauge their feasibility.
- Feedback Integration: Retrieval of feedback from test users (students and instructors) to locate areas of improvement was crucial for the refinement of each scenario.
- Final Design: Optimization of the overall design to ensure the reusability of components was the final hurdle laid out by the team to complete the product.

Key Decision Points & Alternative Solutions:

- Use of VR Technology: Virtual Reality provides an immersive and hands-on learning experience that traditional teaching methods lack. It also offers a more budget-friendly alternative to providing additional education to students that would build upon their theory. Alternatives taken into consideration were the conventional E-learning platforms, AR-based approach, and using a pre-built VR Training environment. An E-learning platform was turned down as it offered a low level of immersion and low engagement that were critical to our goal of improving the student's confidence. The Augmented Reality (AR) approach was rejected as it required a far greater level of resources for half the end reward. It lacked the level of interaction we were investigating to simulate complex cybersecurity events. Lastly, using an existing VR platform that we could then modify was turned down as there was no guarantee that we could customize it to meet our needs as well as the potential cost for licensing and customization.
- Syllabus Architecture: Allowing each cybersecurity topic to be a solo module was done to allow flexibility without hindering the overall system. This is for the reason that adding or changing existing modules would not become difficult while also offering instructors and students the freedom to move at their own pace and help identify where their strengths and weaknesses lie. An alternative idea that was considered was a combined integrated design which was rejected as it was more difficult to maintain and offered less flexibility to the user.
- Hardware Decision: The Oculus Quest 3 was selected based on 2 reasons. The first was in its widespread availability within the university as well as its popularity within the VR market which would lower the percentage of individuals who were unfamiliar with the

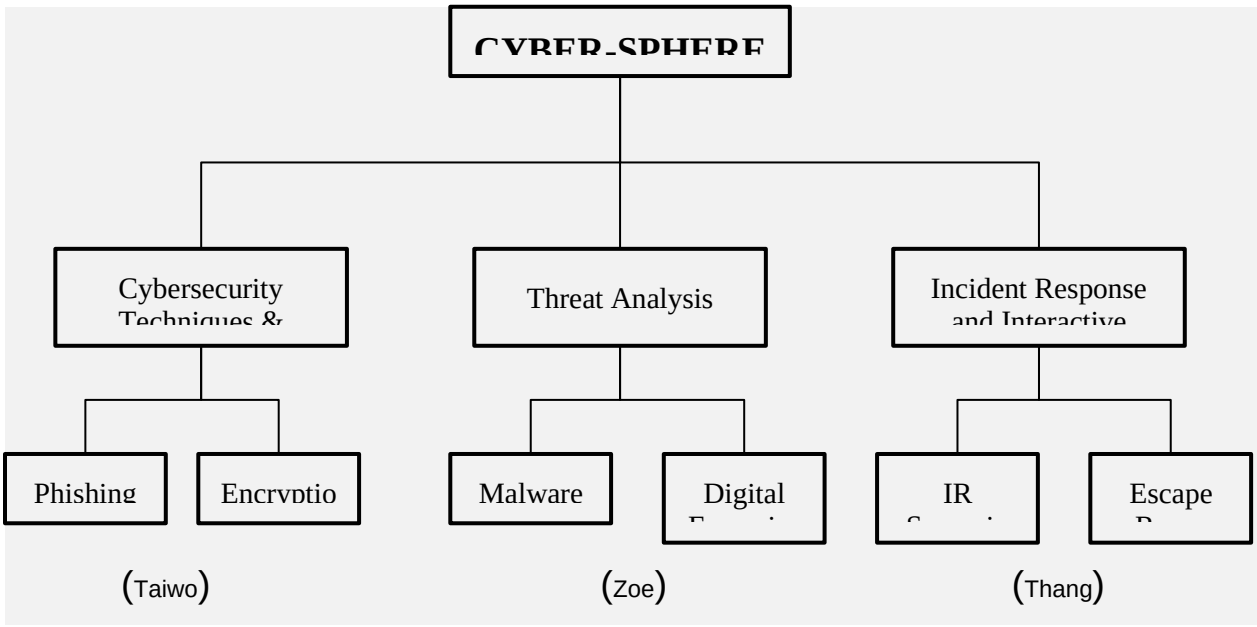
technology and its use. The second was in its accessibility as well as not requiring high-end additional hardware, unlike alternative PC-based VR equipment that we considered such as the HTC Vive Pro.

In conclusion, the final design our team concluded upon was a result of multiple sub-areas of concerns that we believe are being addressed with the most important being ensuring that users are actively engaging, learning, and applying their theory that will lead them towards being better well-equipped industry professionals.

Implementation Details:

Our innovative product was implemented by using Unity and virtual reality settings.

Some problems we received were technical since each team member had difficulties running the Oculus Quest (testing device) on our laptops. This required us to have to constantly make visits to the university to use the computers in the VR room in building 15.



Taiwo: For my section of CyberSphere which focuses on CyberSecurity Techniques and Threats, the implementation involved creating a VR scenario where users learn about phishing attacks, encryption techniques, and information on other cyber threats and techniques used. The goal is to teach students not just about the theory behind these concepts, but also how to spot and counter them in a practical setting.

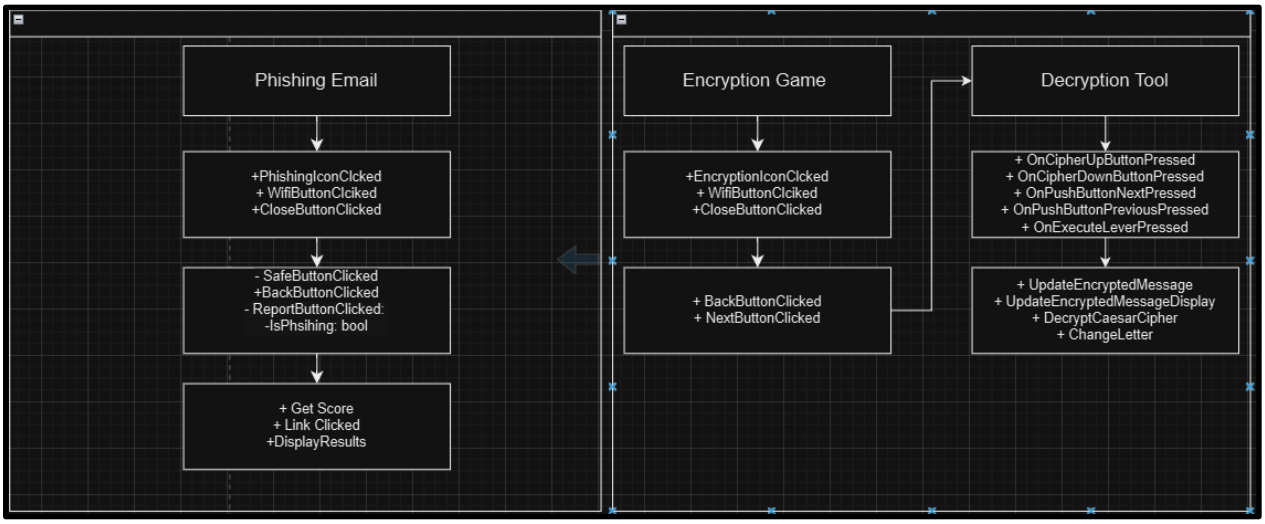
Using the Unity 6 game engine as the main platform for development due to its efficient and easy-to-use features for VR development, the various interactions within the module were made allowing the users to explore the scene, learn, and engage with phishing attempts while also applying encryption/decryption techniques.

Throughout the entire process, there were several challenges I faced during development such as:

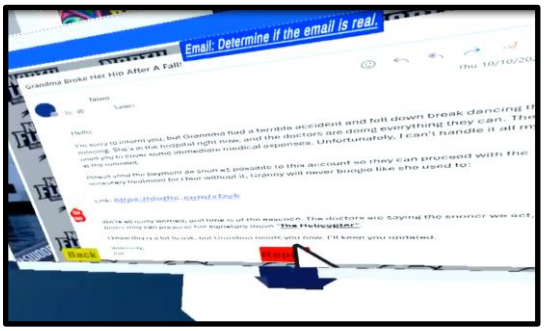
- Device-Related Issue: Using the Oculus Quest 3 headset and controller, ensuring compatibility with the game in Unity was a major issue. While Unity offered great support in developing the VR game, using the Oculus Quest 3 with personal devices such as my laptop for testing was difficult due to system configurations. This, along with the limited testing in the university VR labs at the library, resulted in a slowdown in development time.
    - To address this, the team worked with Professor Kevin Pfiel (Head of the Virtual Reality Lab in the School of Computing) to conduct separate testing sessions providing us with more time for development and addressing our device compatibility issues.
- User Interaction Design: Creating unique controls and interactions for users was also another issue I faced. Simulating real-world systems such as a computer with a working

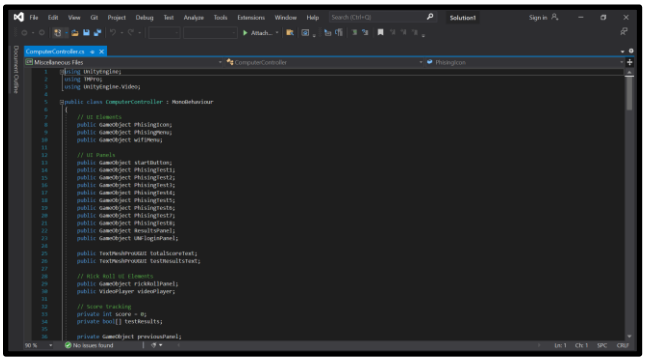
mouse proved to be different due to the transformation of space in VR from the 3D space of the VR environment to the 2D space of the VR computer Screen.

- To address this issue, I made adjustments to Unity's Canvas UI system and wrote a C# script to properly calculate the transformation of mouse movement within the x and z axis to the screens x and y-axis.



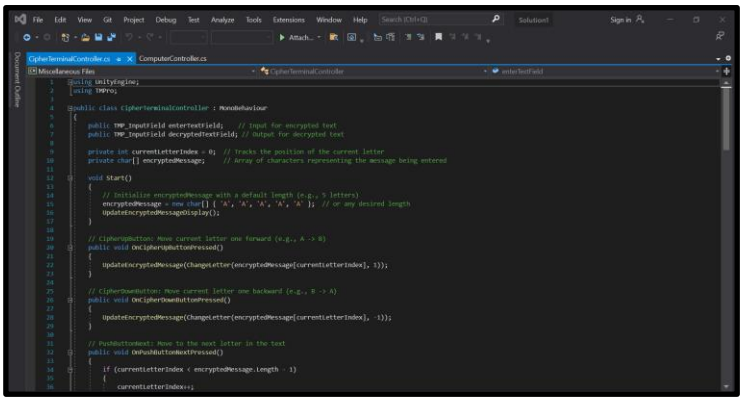
The decision to go with a phishing email simulator was to allow users to interact with an email inbox and identify phishing attempts. The main feature involves selecting emails and analyzing their content for signs that would hint at it being a potential phishing email such as suspicious links or incorrect sender information.





The script provided above serves as the controller for the VR computer, checking if the user's response to the specific email was correct before updating the UI accordingly. When the user uses the cursor to interact with the Phishing Icon, it launches the simulator accordingly.

The Encryption puzzle involves users decrypting a message using a Caesar Cipher decryption tool that is provided to them. The interaction involves users beginning the encryption puzzle by selecting the appropriate icon on the next computer and being presented with a riddle upon which they must decrypt the provided message.



The script provided serves as the controller for the decryption tool. Once users enter the encoded words, the tool returns the decrypted meaning. This interaction helps reinforce the concept of encryption as well as data security.

Zoe': Threat Analysis, which is the section I created, includes both digital forensics and malware. The section allows users to learn about different commands used for digital forensics and also introduces users to malware. For example, the malware that the user will find throughout the simulation is Trojan Horse. There is also a section of the virtual reality

environment that is dedicated to teaching users about Trojan Horse while another section teaches users about malware.

To teach users about digital forensics, they were required to travel between rooms to find commands that they must place on their virtual computer. This would result in them visiting a website that informs the user to download a malware-infected file. The user is then instructed to analyze the image through a virus scanner and download an antivirus to scan their device. Throughout these steps, users can gain skills in threat analysis, digital forensics, and malware. It also simulates a real-world scenario where users must analyze files, images, etc. using the commands (xxd, exiftool, etc.) shown throughout the Threat analysis simulation.

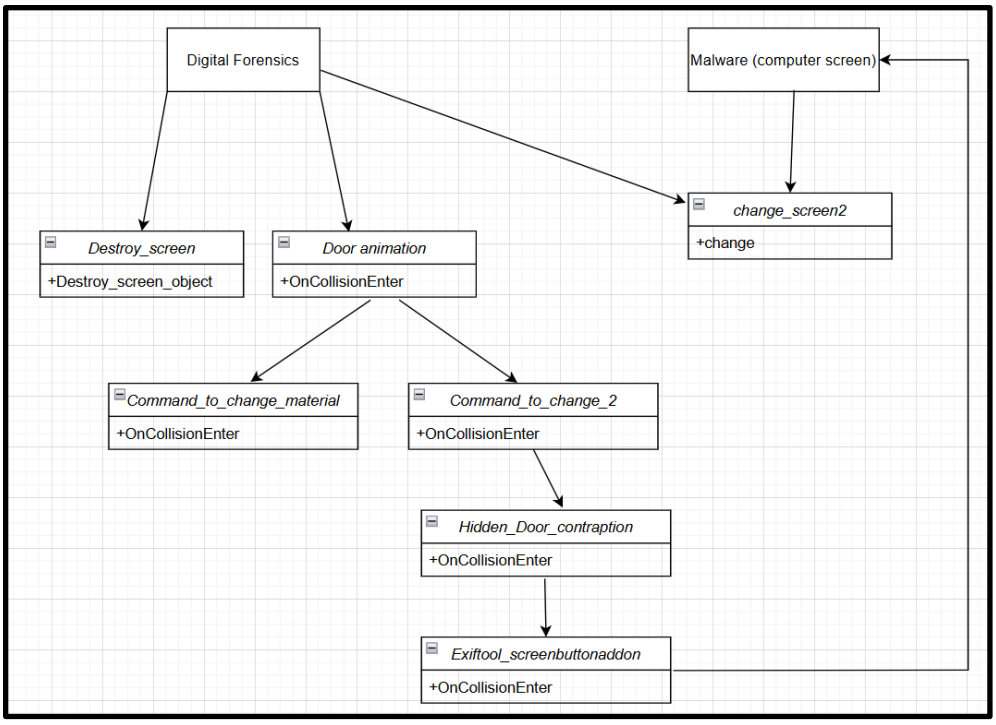
The main platform used for this section was Unity 2022.3.45f1; the platform allowed me to introduce multiple assets and visuals needed for users to learn about Threat analysis. Blender was also used to create different items needed in the virtual environment.

While developing the Threat Analysis section, some risks and difficulties I experienced were:

- Device compatibility: Currently, the laptop that I own is not compatible with the Oculus Quest 3 and Meta Quest link software. Previously, the device seemed compatible but I was informed by Meta that the laptop holds an integrated graphics card which is not compatible with the Meta Quest Link. This issue caused a slowdown in the Threat Analysis development near the beginning of the process since I could not use the Oculus Quest 3.
    - To address the issue, I was able to visit Professor Pfeils VR lab in building 15 to test the product. I could test the product and note specific changes needed before

visiting the lab again. This created a great environment where each team member was able to meet and discuss issues presented to them.

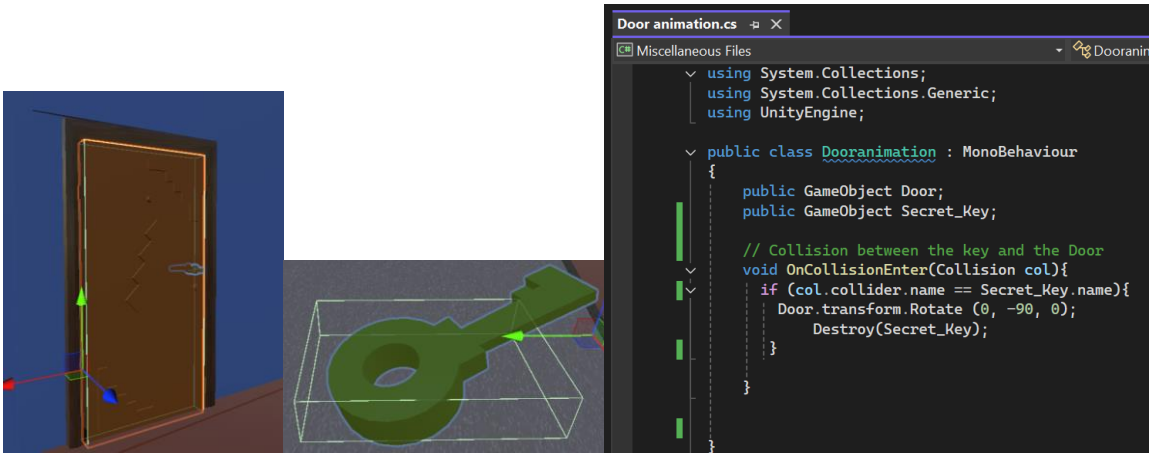
- User interaction with objects: Since during the VR environment users can enter specific rooms depending on the key they hold caused issues since the doors could not identify what key was interacting with the door and would stop the user from entering even if the room was open.
    - To fix this issue, I wrote a C# script that would identify the specific key the user was holding. This allowed the user to enter with ease and not worry about holding the wrong key since each key is color-coordinated with its room.



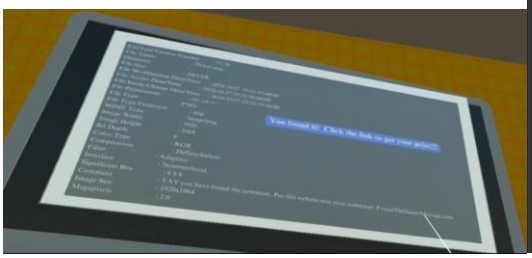
- The UML diagram shows the code used throughout the product. Each section shows a piece of code that was used. I decided to implement the Threat analysis portion in this specific way to represent a real life scenario; by analyzing the image using commands

(digital forensics), it will result in the user receiving a Trojan horse malware on their virtual machine (malware).

- The code was connected by using *Interactive Events* which are interactable objects that users can interact with. Each action is also needed to be conducted to progress through the simulation. For example, the Exiftool_screenbuttonaddon needs to occur to continue to the Malware portion of the simulation.



The image above shows the issue I received between the door and the key. The first image shows one of the three doors users can enter while having the Secret key for that specific room. The highlighted box around the key (second image) and door are the box colliders which is an invisible box that handles collisions with other items; this means the key will collide with the door when their invisible green box collides which automatically allows the code (third image) to run and open the door. This portion provides users access to the rooms with commands; these commands are needed to progress through the simulation.



```
change_screen2.cs
Miscellaneous Files                    change_screen2

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class change_screen2 : MonoBehaviour
{
    public GameObject Button;
    public Image screen;
    public Material material;
    public GameObject NewButton;

    //changes from the virtual machine screen to the website clicked.
    public void change() {
        Destroy(Button);
        screen.material = material;
        NewButton.SetActive(true);

        }
    }
```

` The Malware process starts when the user selects the link hidden in the images information. When clicked, they will be taken to the fake “winning” website. As they continue through the simulation, they will discover that the file they download from this website is infected with Trojan Horse malware.

The script shown above allows the user to navigate through the options presented on the computer screen. Once the user clicks on the designated area on the computer, they will be directed to another section. For example, once the user downloads and opens the file, they will be initiated to exit out of the file and web page where they will enter their virtual machine.

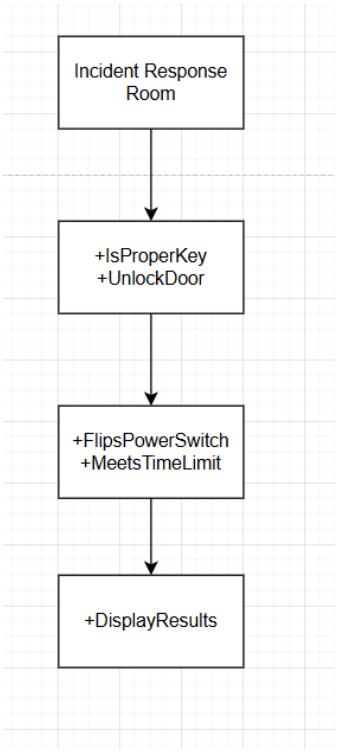
Thang: My particular focus during the development of this virtual reality experience revolved around the concept of Incident Response. Players will be able to experience a scenario where they are responding to an alert given by the system, and if they are unable to respond properly, their system will be compromised. This scenario shows the realistic perspective on how incident response works in the real world, as procedures are followed when responding to any incidents that occur in the workplace.

Throughout the project, several issues arose that plagued the development during this time:

- Device Issues: The Oculus Quest 3 was not compatible with my personal laptop. As such, it made developing the scenario difficult to complete. While there was availability in both the University's VR rooms in the Library and the VR Lab led by Professor Kevin Pfeil, due to my work schedule, time for me to test the scenario was limited to whenever I was able to get to campus on time. I had also attempted to use an alternative method of testing that was available in the Meta SDK All-in-One package, however, that method seemed to also not work when attempting to simulate and test the game. In addition to this, there were two occurrences where my project had gotten corrupted while I was working on the project and I did not backup my project to GitHub at those points, greatly delaying the development of the rooms.
    - To address the initial issue, I relied on the flexible time that was available with the VR lab led by Professor Pfeil, as there were lab assistants who had keys to the room. Due to that being the case, I was able to test my rooms and had to make the most out of my time as much as possible. Meanwhile, for the second issue, as I had already known what asset I would need, so most of my remaining development had focused on fine tuning the assets to fit the environment as well as re-writing any potential scripts needed.
- User Interaction Design: Creating the possible interactions a user can take within the room started to become difficult. This problem was prevalent primarily because incident response revolves around the idea of following a set of procedures, with the procedure

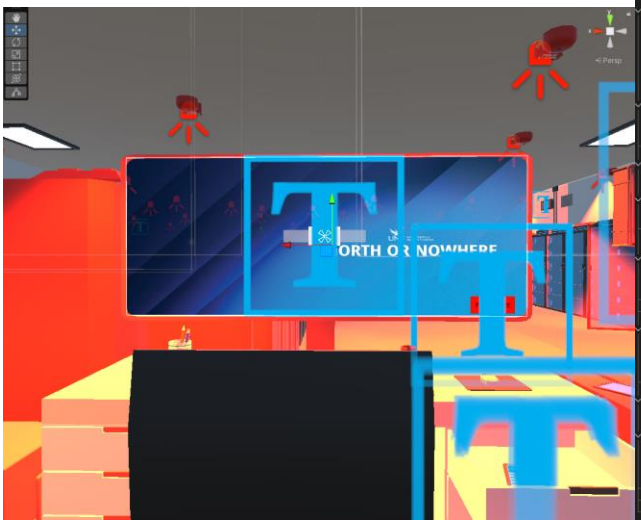
being consistently conducted in all scenarios with some level of flexibility to be able to be applied in multiple scenarios.

- To resolve this issue, I decided to go with a scenario based challenge that would contain the essence of a potential procedure in the scenario itself. This would primarily revolve around following the set of instructions, as well as having an easter egg object that would show a well known incident response framework that is commonly used to structure an incident response procedure.

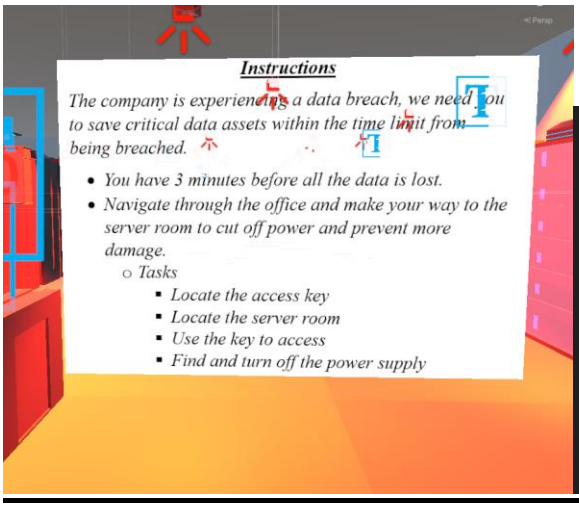


The UML diagram shows the intended path of the incident response room. Upon entering the first room, players would be presented with a set of instructions that details the ongoing incident that is occurring and detailing what the next steps should be taken to complete the challenge. Originally, there would have been an interactive computer for the user to go through step by step. Unfortunately, the testing was unable to fix the issue and we went with a simple canvas that showed the instructions directly. The first two photos below show the original

attempt and the last two photos below show the final product along with the flashing lights to signal the urgency of the scenario.



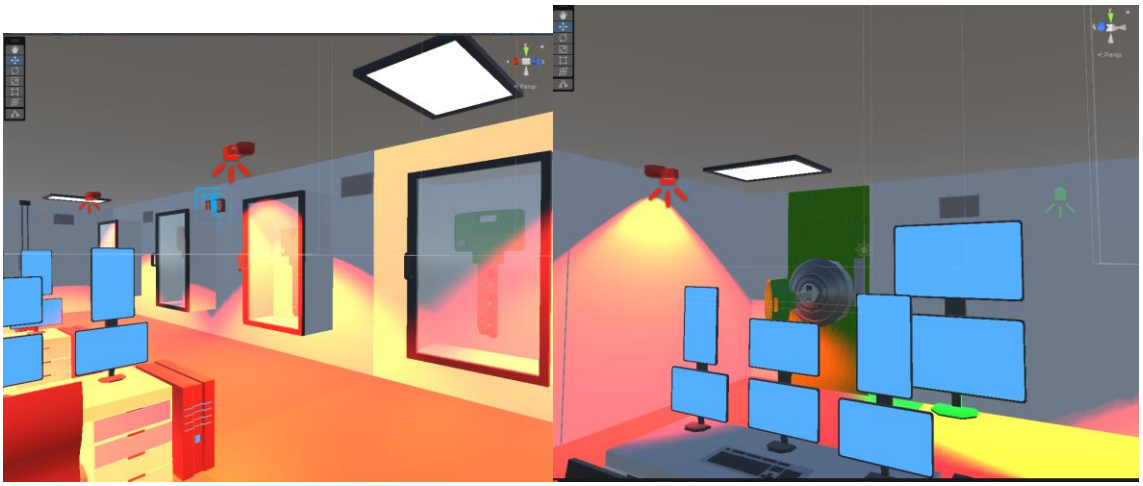
```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class CursorController : MonoBehaviour
{
    public GameObject mouse;
    public float sensitivity;
    public XRBaseController controller; // Reference to your VR controller for input
    private XRDirectInteractor interactor;
    private XRBaseInteractable hoveredInteractable;

    void Start()
    {
        interactor = GetComponent<XRDirectInteractor>();
    }

    void Update()
    {
        // Update cursor position based on 3D mouse movement
        transform.localPosition = new Vector3(-mouse.transform.localPosition.x * sensitivity,
                                              -mouse.transform.localPosition.z * sensitivity,
                                              transform.localPosition.z);

        // Only trigger SelectEntered on click input
        if (hoveredInteractable != null && controller.selectInteractionState.activatedThisFrame)
        {
            interactor.StartManualInteraction(hoveredInteractable);
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        // Detect when cursor enters button area
        hoveredInteractable = other.GetComponent<XRBaseInteractable>();
    }

    private void OnTriggerExit(Collider other)
    {
        // Clear the interactable and end interaction on exit
        if (other.GetComponent<XRBaseInteractable>() == hoveredInteractable)
        {
            hoveredInteractable = null;
            interactor.EndManualInteraction();
        }
    }
}
```



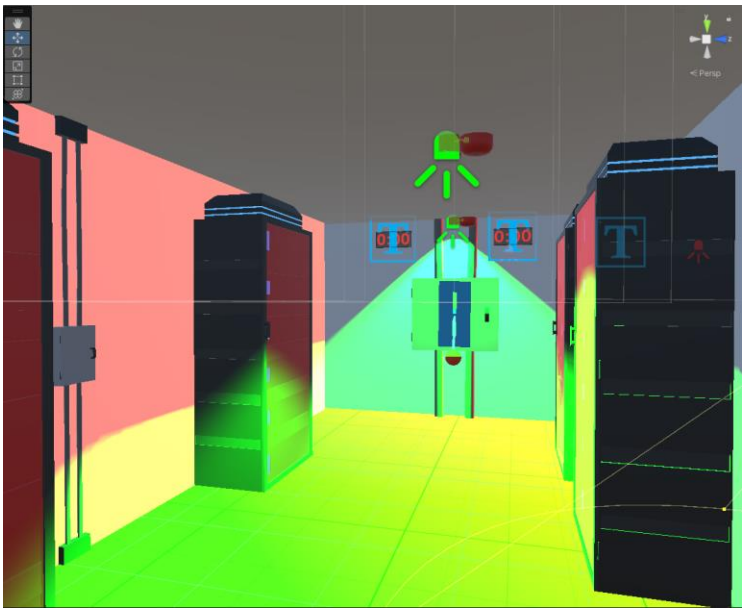
Miscellaneous Files — FlashingLight

```
using UnityEngine;

public class FlashingLight : MonoBehaviour
{
    public Light lightSource;  // Assign your light in the Inspector
    public float flashSpeed = 1f;  // Speed of flashing (higher = faster)

    private bool isFlashing = true;

    void Update()
    {
        if (isFlashing)
        {
            float intensity = Mathf.PingPong(Time.time * flashSpeed, 1f);
            lightSource.intensity = intensity;
        }
    }

    public void ToggleFlashing(bool flashing)
    {
        isFlashing = flashing;
        if (!flashing)
        {
            lightSource.intensity = 0f; // Turn off light when not flashing
        }
    }
}
```

Following that, the user would go into the room and attempt to find the correct key that would then unlock the door to the server room.



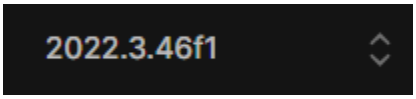
Finally, the user would then turn off the power switch to pass if they had done it in the proper amount of time or fail if the player did not complete the challenge in time, which is shown below.



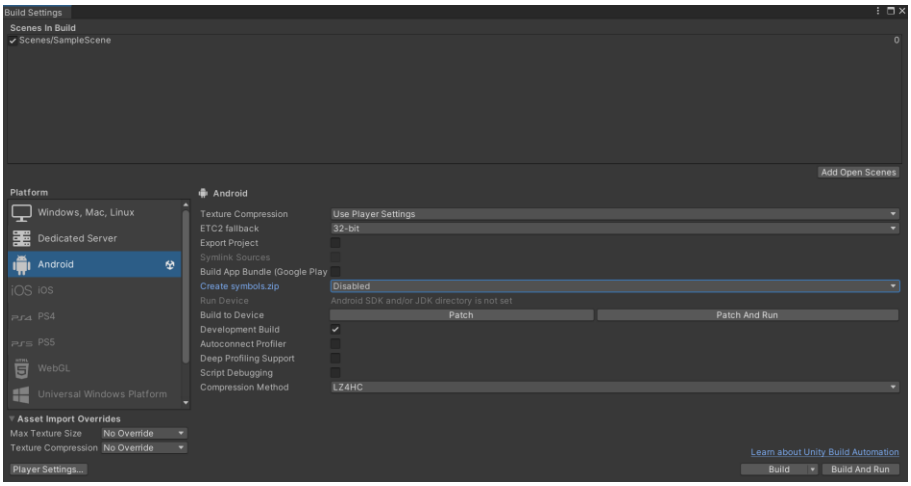
Product Tutorial:

To use our product, you must have the correct version and platform to run the program. Each of the files we provide utilizes different versions of Unity. Users must first install Unity Hub, Meta Quest Link, and the correct Unity version affiliated with the product version. Users must obtain an Oculus Quest headset and create a meta account if they would like to test our product. After creating their accounts, they must download the meta app also to their phone and turn “developer mode” on.

Example image of version:



Example of how to build and run our product:

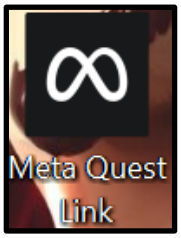


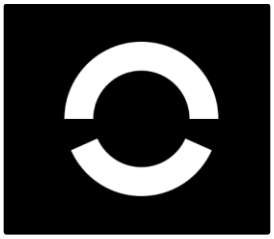
During the build and run process, users will be provided with the option to change to the Oculus Quest 3. They must then select Build and Run to run our product.

<u>Taiwo</u>: To commence the “Phishing & Encryption” VR game, the following steps must be followed:

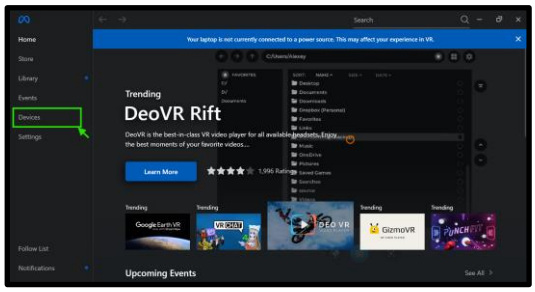
- Have the following equipment: An Oculus Quest 3 headset, controller, USB type C cable, a Laptop, or desktop computer, and a smartphone.

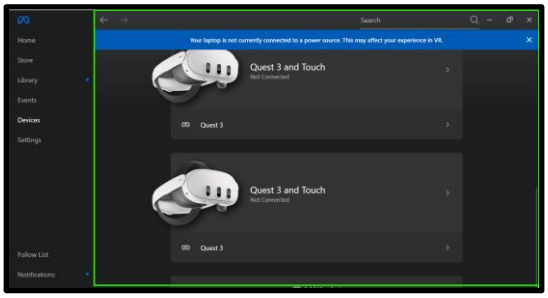
- Have the following software installed on your PC: Unity Hub and Meta Quest Link. Have Meta Horizon installed on your smartphone.



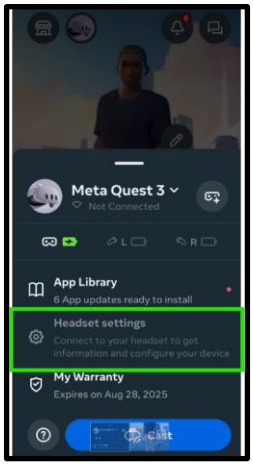


- Power on your Oculus Quest 3 headset and connect the USB cable to your pc > Open the Meta Quest Link on your pc >Sign in or create an account.
- Select devices > set up the configuration for your quest 3



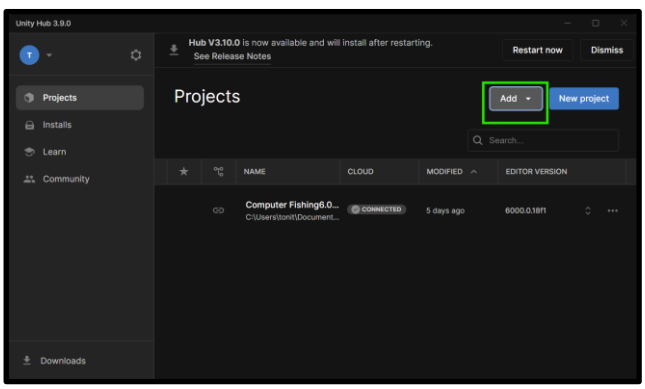


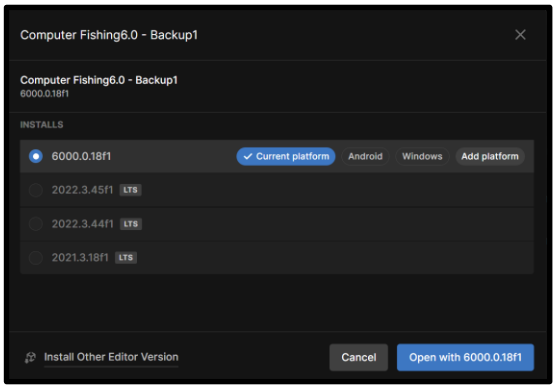
- On your Oculus headset, ensure it is connected to the same wifi network as your smartphone.
- Open the Meta Horizon app on your smartphone > Sign in to the same account. > Navigate to devices > Select your Meta Quest 3 device > Headset Settings > Navigate to developer mode and enable it.



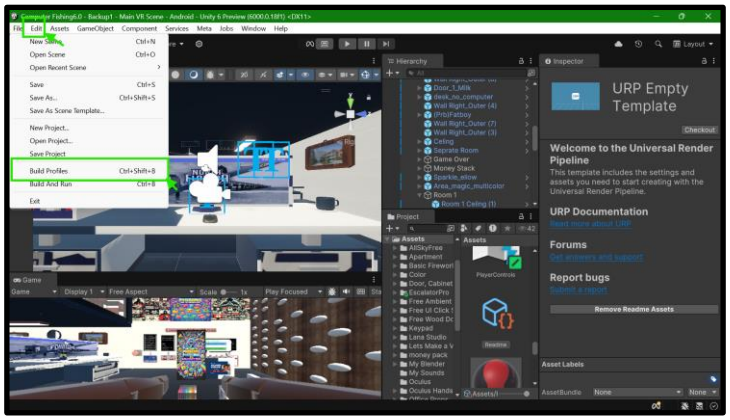
- On your Oculus Quest, you should receive a notification to allow debugging > select allow.

- Open the Unity Hub > > Add > Add Project from disk > Navigate to the location of the game file "Computer Fishing6.0 - Backup1" > launch the project (Note, you may be required to install Unity 6 editor before launching, if so, install it along with the Android and Windows JDK platforms).

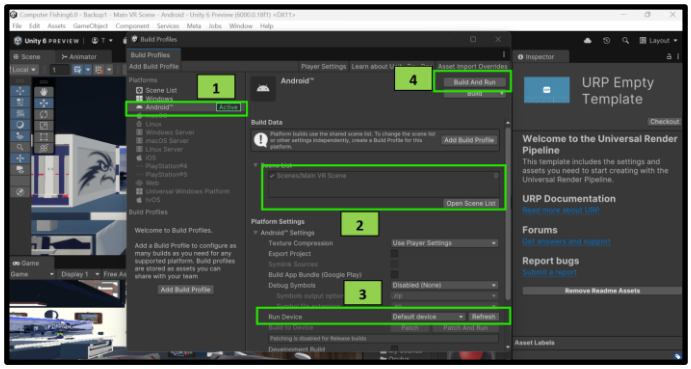




- Once the project launches, navigate to the file (top left corner) > build profile.

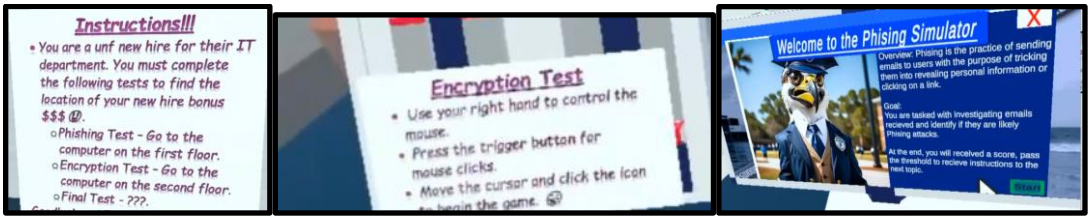


- Select Android as the platform to run (if an option to switch platforms appears, do so) > Ensure the scene is set to “Main VR Scene” > On Run Device, select refresh if the Oculus Quest 3 headset is not displayed on the list then select it (if the oculus does not show up, turn off and back on developer mode on your smartphone, then allow USB debugging on your headset) > build and run > select an appropriate location to store the build file.

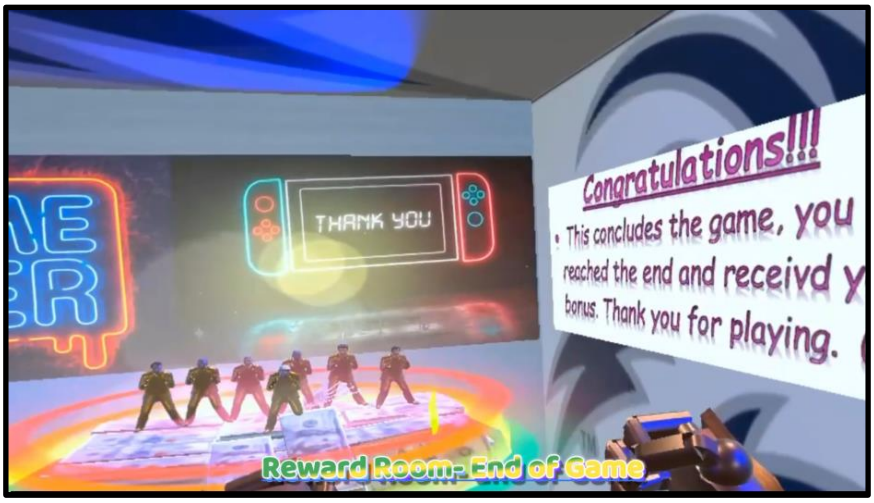


- The Phishing and Encryption game should load on your VR headset after a slight delay.

- Instructions are present in the game that will guide you on tasks required to complete each scenario.



- Upon completing the final step, the user will arrive at the location of their reward thus completing the game simulation.



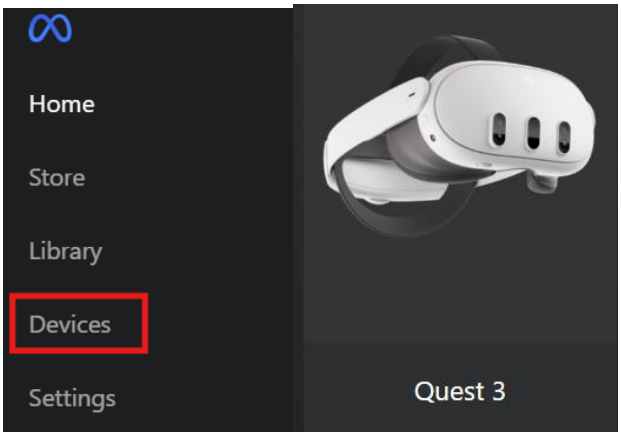
Zoe': The user must take the steps listed carefully to be able to run the product. The device (computer/laptop) they will be using must be compatible with the Oculus Quest 3. If not, they must obtain a different compatible device They must have:

- Compatible laptop/computer
    - Software needed to be downloaded
        - Unity hub
        - Meta Quest link
- Oculus Quest 3
- USB A to USB C cable (compatible with Oculus Quest 3)

- Mobile Device
    - Software needed to be downloaded
        - Meta Horizon app

To run the Threat Analysis VR game, you must follow each of the steps below and have the required equipment listed above.

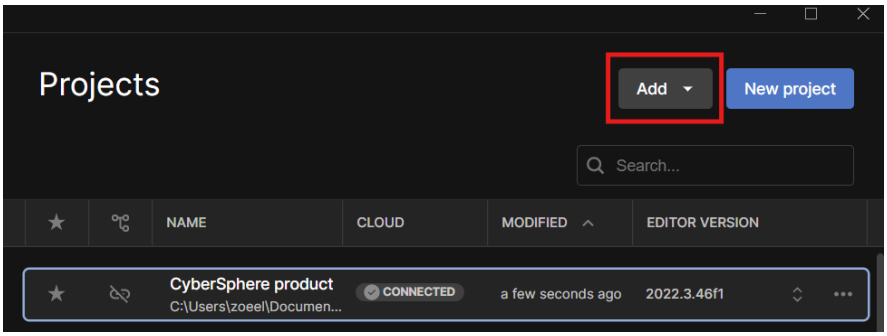
Meta Quest Link (Step 1):



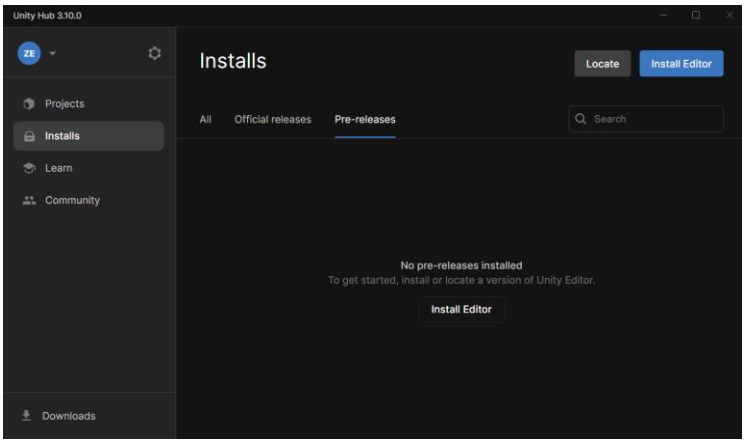
To test this product, the user must first download the Meta Quest link. They will be asked to create an account or log in. Once they are logged in, they must select devices and add Quest 3 to their account. The Oculus Quest 3 must be on and connected to the device by using the USB A to USB C cord to run. The Oculus Quest 3 must also be connected to the same Wi-fi as your computer/laptop.

Meta Horizon app (Step 2): The user will need a Meta Quest account to log into the meta horizon app. Once they are logged in they must first select the picture of the Oculus Quest 3 (shown on the bottom left). They will then be provided the options for the Meta Quest 3 where they must enable developer mode through the Headset settings. This app also allows users to cast videos from their headset to their phones if needed.

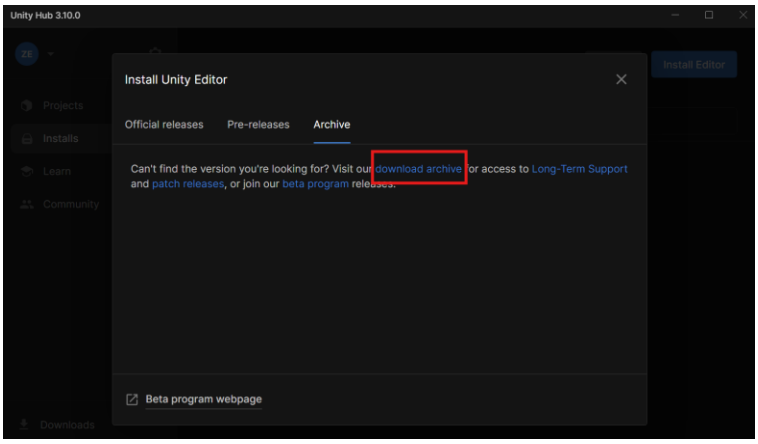
Adding version & project (Step 4):



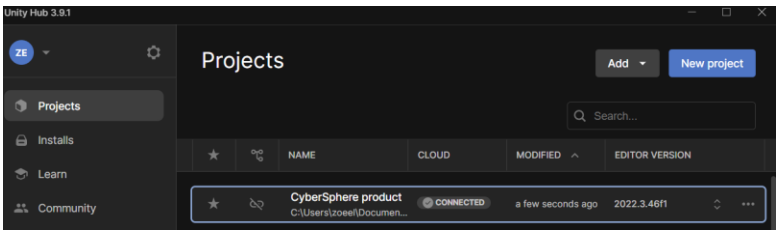
- These actions will be taken in Unity Hub assuming the user has a unity account to continue. If not, they must create one. Next, you should select Add and then select the folder with the product. Verify that the project folder is not inside of another folder when adding. You will receive an alert after placing the folder since the *Editor Version* has not been installed. To start installing the *Editor Version*, select *Installs*.



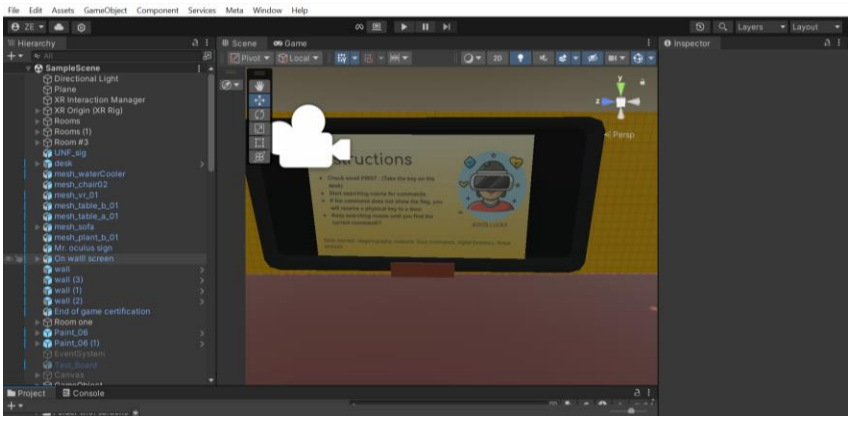
- Next, select Pre-releases and then Install Editor.



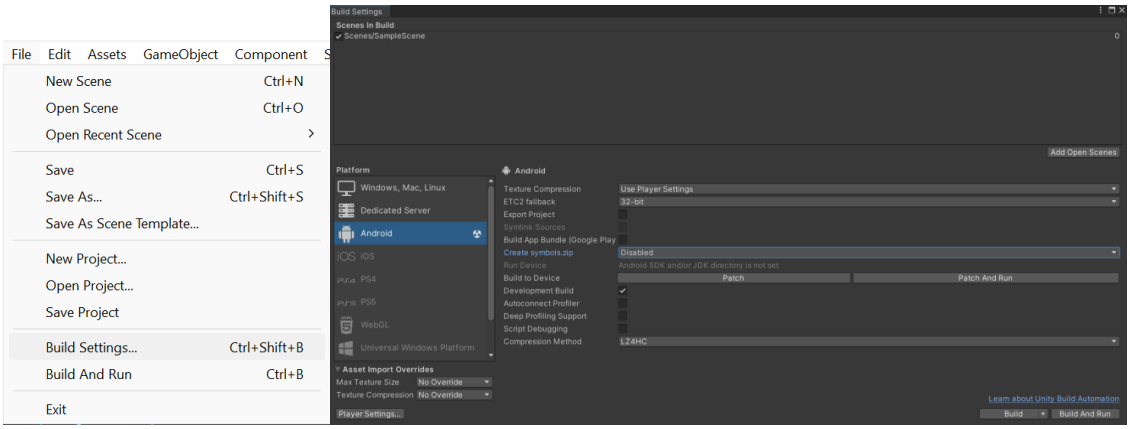
- The *Editor Version* being used is not installed so you must install 2022.3.46f1



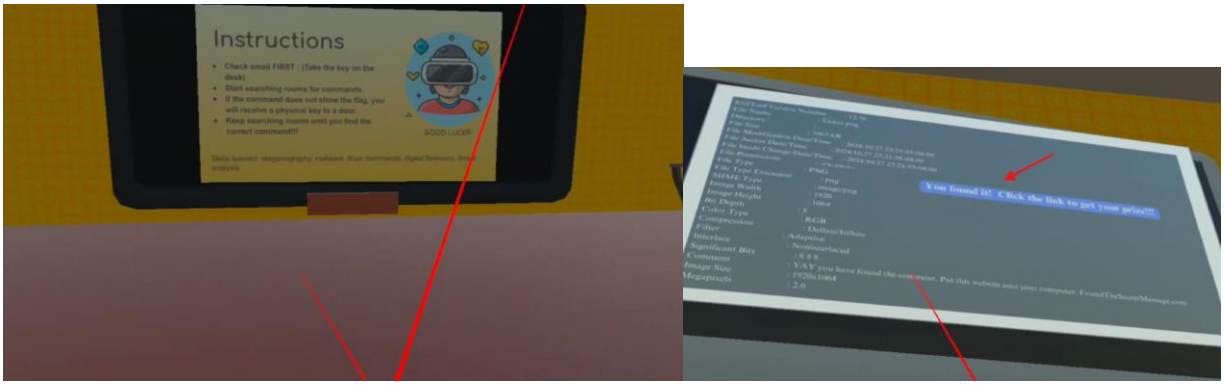
- Once they have downloaded the version and have imported the folder, they will select the folder in Unity Hub to run the program.



- They will be taken to the screen shown above. To run the folder, they must first select the file option (top left corner of the screen).



- On the top left corner (first image), they will be provided with this list of options. They must select Build Setting which will take them to the build and run screen. They will be provided the option to change their device to the Oculus Quest 3 as long as they have the Oculus connected to their computer/laptop with a USB A (connected to the computer/laptop) and USB C (connected to Oculus Quest 3).
- On the top right image, users will be provided with the option of Oculus Quest 3 only if the device is connected correctly. If the device does not appear, you must repeat the previous steps for the Oculus Quest 3 connection. If the device is connected, the user will instead be provided with an option to change their device under *Run Device*. After the Oculus Quest 3 is selected, you must select *Build And Run*.



- Instructions are the first object seen by the user when entering the game. It explains to the user what they will need to do to complete the game. Further instructions are given to the

user when they are interacting with the computer; for example, image two shows how the user will receive instructions while progressing through their virtual personal computer.

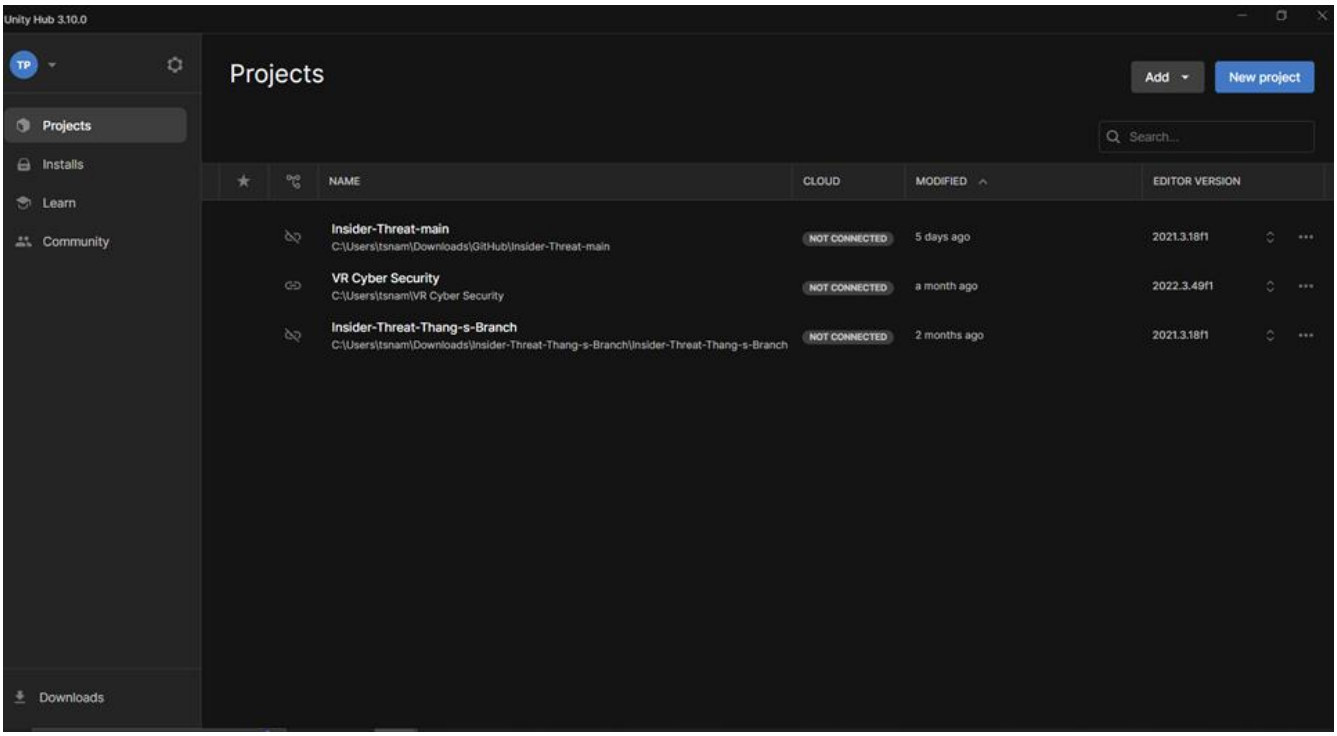
- The final room provides the user with their simulation certificate after completing all tasks.

Thang: To run the incident response simulation, the user must meet several requirements on their machine to ensure that their machine is capable of running the simulation. The first would be to ensure that the machine the player uses, whether it is a computer desktop or a laptop, is compatible with utilizing the Oculus Quest 3. If the machine is unable to support the Meta Quest 3, the user will have to find a different machine that is capable of doing so. The following equipment and software must be acquired before running the game:

- Compatible computer desktop or laptop
    - Required Software:
        - Unity Hub
        - Meta Quest Link
- Oculus Quest 3

- USB-A to USB-C cable
    - The USB-C end will be connected to the VR headset
- Mobile Device
    - Required Software:
        - Meta Horizon app
- Setup:
    1. Meta Quest Setup – Link
        a. Once the Meta Quest Link has been installed onto the desired machine, create an account or log in with an existing account.
        b. Select the “Devices” tab on the left-hand column and set up your headset.
        c. Ensure that the headset is connected via the USB A to USB C cable, with the USB C end connected to the headset. Make sure the headset is on the same network as the desired machine.
    2. Meta Quest Setup – App
        a. Create a Meta account or log in with an existing account to the Meta Horizon app on your phone.
        b. Navigate to Devices → Select the icon for your Meta Quest 3 device → Headset Settings → Developer mode → Enable it
    3. Unity – Hub
        a. Once Unity Hub has been installed, log in with an existing Unity account or create one.

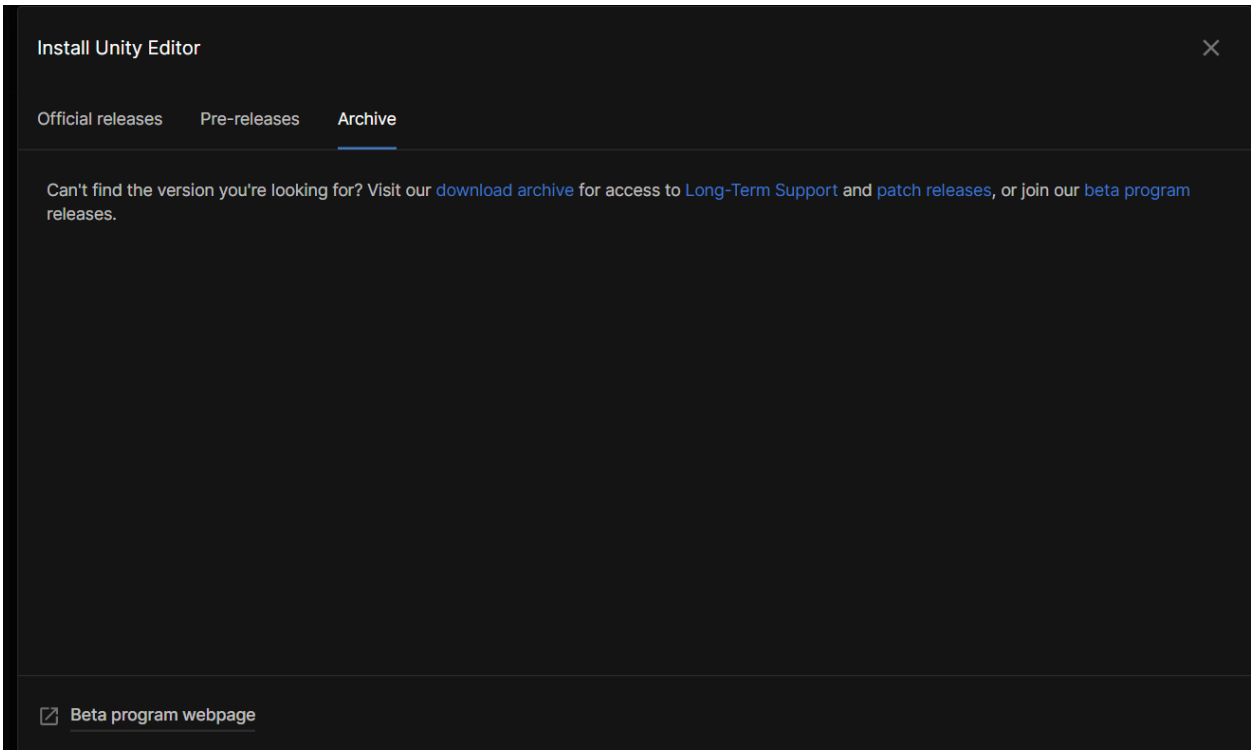
b. In the top right of your Unity Hub home, in the “Projects” tab, select the “Add” dropdown, as below, and navigate to where you saved the folder project. Select the folder you wish to add to your projects.



c. Install the required Unity Editor version to run. This editor version requires version 2021.3.18f1.

d. Navigate to Installs → Install Editor → Archive

e. In the “Archive” tab, select the link to the “Download archive”.



f. Search for the version to download and select the blue “Install” button next to the version. A pop-up will open and select “Open Unity Hub”

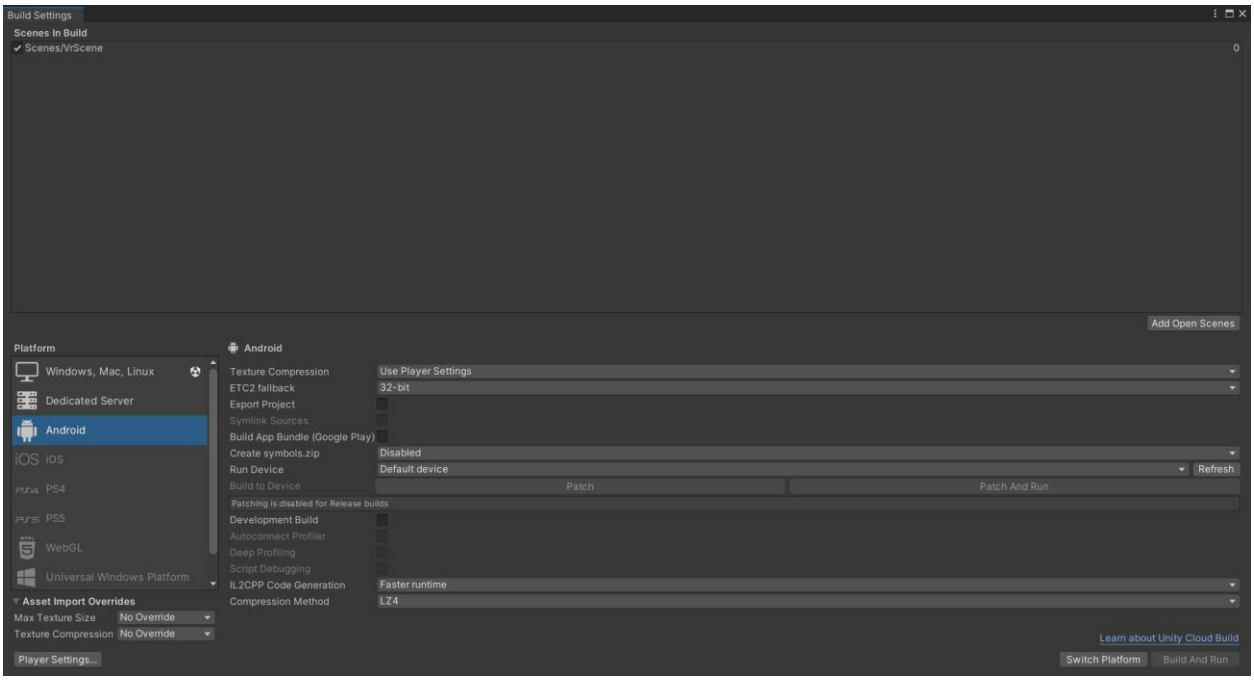
g. Select the Android JDK platform and install it.

h. Once installed, double-click the project name and open the project.

4. Unity – Game

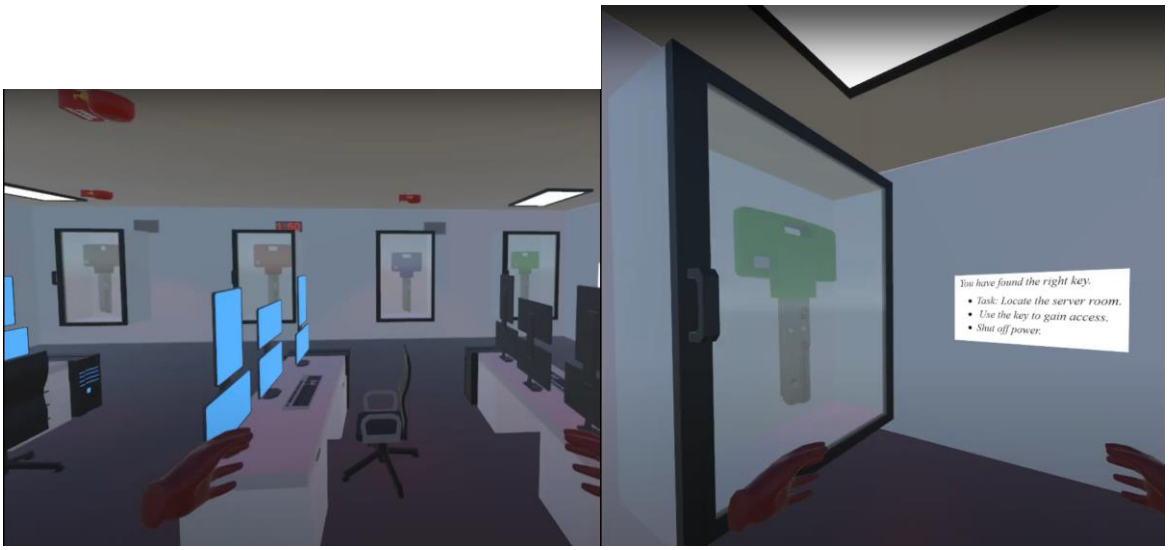
a. Once the file is opened, navigate to File → Build Settings.

b. In the “Build Settings”, ensure that the “Android” option is selected as the “Platform”. Once the Oculus Quest 3 has been connected to the desired machine, ensure that the “Run Device” setting is set to use the Oculus Quest 3 to ensure that the game will port over to the headset when running the game. If the device does not populate in the dropdown option, the player must redo the setup for the Oculus Quest for the machine.



- Playing the Game
    - The user will run into the building with flashing red lights. Upon entering, there will be an initial set of instructions to introduce the scenario to the player and their goal. Upon reading the instructions in the first room, the user will be allowed to explore their area to complete their tasks.

- Once the players arrive in the second room, they will be presented with a wall of keys and the challenge of attempting to find the correct key that will open the door to the server room with the power switch.



- Once they have found the correct key, the player will then bring the selected key, and insert it into the area of the key. The key will be inserted and the door will be opened.

- Upon entering the room, the player will be presented with a server room that contains the power switch. If you can get the power switch on time, a congratulatory message will pop up on the player's screen. But if they are unable to get it on time, they will instead get a try again message. To try again, the players will have to stop the game and rerun it to restart the game.

Evaluation:

Taiwo: To evaluate the usefulness of the Cybersecurity Techniques and Threats module, focused testing sessions with students and professors whom are not fully familiar with cybersecurity concepts were performed. The evaluation aimed to determine whether the module effectively taught users how to identify phishing attempts as well as understand the basics of encryption and decryption methods. With the original problem being the lack of practical and engaging tools in cybersecurity education, an evaluation was done based on the following:

- Users needed to be able to correctly identify at least 60% of the presented emails.
- Users should be able to decrypt the provided messages within a reasonable amount of time.
- A pre-test and post-test assessment is used to compare users' performance in identifying phishing emails and the time taken to solve the encryption module.

Key data collected during evaluation would be made up of the number of phishing emails correctly identified, time taken to complete each email (Phishing), time taken to decrypt an entire message (Encryption), number of hints used by the user to complete the associated problem, number of user input errors, and the overall duration of the session. The data used for the evaluation involved:

Dependent Variable	Accuracy in locating phishing attempts
Dependent Variable	Rate of success in decrypting entire messages
Dependent Variable	User’s level of confidence (1- 5 scale)
Independent Variable	User’s prior knowledge of cybersecurity
Independent Variable	Complexity of module tasks

Data Collection (Quantitative)	Error Count and Success Rate
Data Collection (Qualitative)	User feedback and satisfaction survey

The evaluation and data drawn will be used to refine future versions of the module as well as show how it enhances a user's ability to recognize phishing and understand encryption.

Zoe': To evaluate the utility of the Threat Analysis module, the module was tested with individuals who do not have experience in security. Throughout the semester, the product was shown to individuals who did not have much security experience and asked whether they could understand the information; for example, receiving feedback on whether users were able to understand the definitions presented in the command rooms, etc. Since our problem is to improve the issue of not having enough practical or engaging cybersecurity-related content for cybersecurity education, the Threat Analysis module must have interactive rooms, puzzles, and facts that the user can be able to retain. The evaluation was conducted based on the following:

- Users should be able to open all doors and puzzles to progress through the simulation.
- Users ability to navigate through the simulation and arrive at the final door.
- The amount of time a user takes to complete the simulation
- Users should be able to access each step during the malware section (virus scanner, antivirus, etc.).

The main data collected from testing would be the users ability to travel through each room. The first part of the simulation requires users to travel through rooms and complete puzzles. These puzzles were created to enhance users' ability to retain information learned through the simulation. The amount of time users take to complete the simulation will depend on the amount of knowledge they have about security.

Dependent variable	The amount of time a user replays the simulation
Independent variable	Amount of security knowledge known prior
Independent variable	Previous Knowledge of Security
Data collection (Quantitative)	External timer (phone, clock, etc.) when traversing the simulation
Data collection (Qualitative)	User Feedback & Surveys

The evaluation will be used to make future changes to our design by adding new modules, sections, and possibly, levels (beginner, advanced, etc.).

Thang: To evaluate the utility of the Incident Response module, the module was tested with individuals who have little to no understanding of cybersecurity. Throughout the semester, the module was presented to a wide range of audience for them to attempt to go through the challenge room and check to see if they were able to comprehend the lessons being taught in the game. In this case, would the users, upon going through the game, recognize that there was a procedure to the scenario and the severity that is associated with responding to an incident. As our originals intentions with the game focused on increased knowledge retention by utilizing virtual reality as a platform, the room would require there to be some level of freedom, interaction with the items in the room, but also a time constraint due to the sensitivity of responding to an incident in real time, while maintaining the service level agreement that an analyst would meet in the real world. The room tested the following conditions:

- Users would be able to complete the challenge in a timely manner
- Users would be able to utilize the tools given to them by the challenge room effectively

- Users would be able to respond appropriately given the proper amount of information in the instructions

Data collected during evaluation would consist of the number of attempts to complete the simulation, the length of time remaining to complete the challenge, the user's baseline knowledge in regards to cybersecurity, and timer used to check the length of time it took to complete the simulation.

Dependent Variable	Number of attempts to complete the simulation
Dependent Variable	The limited timer time for the user to complete the simulation
Independent Variable	User's baseline knowledge of cybersecurity
Data Collection (Quantitative)	Timer used to check against user when running the simulation
Data Collection (Quantitative)	User Feedback & Surveys

The acquired data will be used to further tune our existing scripts, create new features, and increase difficulties for future versions of the simulation.

Conclusion:

CybersSphere addressed the key problem of outdated and theoretical teaching methods in cybersecurity education. By leveraging Virtual Reality technology, we created an innovative product that offers a hands-on and interactive experience in three major areas: Cybersecurity Techniques and Threats, Threat Analysis, and Incident Response and Iterative Training. Each module provided users with practical skills and knowledge to face real-world cybersecurity obstacles, bridging the gap between theoretical concepts and practical use.

Our design was influenced by existing educational tools and training modules. By taking the best parts from prior research in VR learning environments, we tailored them into a cohesive unit that maximizes user engagement and knowledge retention. Our design process also offers continuous improvement through user testing and scenario refinement ensuring its use matches the users' skill and area of improvement.

The evaluation showed that users significantly improved their understanding of cybersecurity topics after using CyberSphere. For example, the techniques and threats module showed an increase in a users' ability to identify phishing attempts as some users reported being unaware of inspecting emails for incorrect contacts or frequent grammatical errors. These outcomes further validated the product's usefulness and its potential to fill the vital gap in cybersecurity education.

Our product development spanned several months with key milestones including the initial investigation, conceptual design, prototyping, and final optimization. The biggest challenge our team faced was the integration of our project into the Oculus Quest hardware and ensuring its compatibility. CyberSphere achieves its set goals, regardless, some limitations remain such as:

- Scalability for larger user groups.
- Limited resources that restrict going outside a certain scope.
- Lack of integration with other educational tools/platforms.

By demonstrating the value of VR-based education in a highly technical field, CybersSphere contributes significantly to the existing body of knowledge. It provides an easily replicable model for developing immersive educational tools that can be used in other fields of

STEM. Organizations and Industries can also benefit by incorporating CyberSphere into their training programs to enhance employee training. For future work, we offer the following:

- Enhancing the platform's scalability for multi-user environments.
- Incorporating AI feedback framework for dynamic learning paths.
- Extension of compatibility features to other VR hardware as well as non-VR platforms for wider accessibility.

In conclusion, CyberSphere shows how an innovative product can redefine education, providing users with an engaging and practical way to improve cybersecurity. By furthering knowledge, improving skill development, and tackling real-world needs, this project adds a meaningful impact to the academic and professional communities.

References:

- Abdullah M. Alnajim et al. 2024. *Exploring Cybersecurity Education and Training Techniques: A Comprehensive Review of Traditional, Virtual Reality, and Augmented Reality Approaches*. Retrieved December 12, 2024, from https://go.gale.com/ps/i.do?p=AONE&u=jack91990&id=GALE%7CA779344500&v=2.1&it=r&sid=bookmark-AONE&asid=457b0de6.

- Silvestro V. Veneruso, Lauren S. Ferro, Andrea Marrella, Massimo Mecella, and Tiziana Catarci. 2020. *CyberVR: An Interactive Learning Experience in Virtual Reality for Cybersecurity Related Issues*. In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, Springer, 89–100. DOI: 10.1145/3399715.3399860.

- Alnajim, A. M., Habib, S., Islam, M., AlRawashdeh, H. S., & Wasim, M. (2023). Exploring cybersecurity education and training techniques: A comprehensive review of traditional, virtual reality, and augmented reality approaches. Symmetry, 15(12), 2175. DOI: https://doi.org/10.3390/sym15122175.

- Chow, Y. W., & Susilo, W. (2024). Strategic approaches to cybersecurity learning: A study of educational models and outcomes. Information, 15(2), 117. DOI: https://doi.org/10.3390/info15020117.

- Chittimalli, P., & Satyam, S. (2023). Immersive technologies for cybersecurity skill development: An analysis of virtual and augmented reality applications. ACM Transactions on Computing Education, 23(1), Article 19. DOI: https://doi.org/10.1145/3583452.

Appendix:

Cybersecurity Techniques & Threats:

Computer Controller Script

```
using UnityEngine;
using TMPro;
using UnityEngine.Video;

public class ComputerController : MonoBehaviour
{
    // UI Elements
    public GameObject PhisingIcon;
    public GameObject PhisingMenu;
    public GameObject wifiMenu;

    // UI Panels
    public GameObject startButton;
    public GameObject PhisingTest1;
    public GameObject PhisingTest2;
    public GameObject PhisingTest3;
    public GameObject PhisingTest4;
    public GameObject PhisingTest5;
    public GameObject PhisingTest6;
    public GameObject PhisingTest7;
    public GameObject PhisingTest8;
    public GameObject ResultsPanel;
    public GameObject UNFloginPanel;

    public TextMeshProUGUI totalScoreText;
    public TextMeshProUGUI testResultsText;

    // Rick Roll UI Elements
    public GameObject rickRollPanel;
    public VideoPlayer videoPlayer;

    // Score tracking
    private int score = 0;
    private bool[] testResults;

    private GameObject previousPanel;
    private GameObject currentPanel;
```

```
private void Start()
{
    testResults = new bool[8];
    BackButtonPressed();
}

// ========================= Button Click Handlers =========================
public void ReportButtonClicked(int testIndex)
{
    HandleReportButton(testIndex, GetCurrentPanel(), GetNextPanel(testIndex));
}

public void SafeButtonClicked(int testIndex)
{
    HandleSafeButton(testIndex, GetCurrentPanel(), GetNextPanel(testIndex));
}

public void BackButtonClicked()
{
    if (previousPanel != null)
    {
        currentPanel.SetActive(false);
        previousPanel.SetActive(true);
        currentPanel = previousPanel;
    }
    else
    {
        PhisingMenu.SetActive(true);
    }
}

private GameObject GetCurrentPanel()
{
    return currentPanel;
}

private GameObject GetNextPanel(int testIndex)
{
    // Logic to determine the next panel based on the test index
    switch (testIndex)
    {
        case 0: return PhisingTest2;
        case 1: return PhisingTest3;
```

```
            case 2: return PhisingTest4;
            case 3: return PhisingTest5;
            case 4: return PhisingTest6;
            case 5: return PhisingTest7;
            case 6: return PhisingTest8;
            case 7: return ResultsPanel;
            default: return null;
        }
    }

    // ========================= Handle Report and Safe Buttons =========================
    private void HandleReportButton(int testIndex, GameObject currentPanel, GameObject nextPanel)
    {
        if (testIndex == 0 || testIndex == 4 || testIndex == 6 || testIndex == 7)
        {
            score++;
            testResults[testIndex] = true;
        }
        else
        {
            testResults[testIndex] = false;
        }
        GoToNextPanel(currentPanel, nextPanel);
    }

    private void HandleSafeButton(int testIndex, GameObject currentPanel, GameObject nextPanel)
    {
        if (testIndex == 1 || testIndex == 2 || testIndex == 3 || testIndex == 5)
        {
            score++;
            testResults[testIndex] = true;
        }
        else
        {
            testResults[testIndex] = false;
        }
        GoToNextPanel(currentPanel, nextPanel);
    }

    // ========================= Go To Next Panel =========================
    private void GoToNextPanel(GameObject currentPanel, GameObject nextPanel)
```

```
    {
        currentPanel.SetActive(false);
        previousPanel = currentPanel;
        nextPanel.SetActive(true);
        this.currentPanel = nextPanel;

        // Check if the next panel is the ResultsPanel
        if (nextPanel == ResultsPanel)
        {
            DisplayResults(); // Call the method to display results
        }
    }

    // ========================= Display Results ========================
    public void DisplayResults()
    {
        totalScoreText.text = "Total Score: " + score;
        Debug.Log("Total score text set to: " + totalScoreText.text);

        string results = "";
        for (int i = 0; i < testResults.Length; i++)
        {
            results += "Test " + (i + 1) + ": " + (testResults[i] ? "Correct" : "Incorrect") + "\n";
        }
        testResultsText.text = results;
        Debug.Log("Test results text set to: " + testResultsText.text);

        string passFailMessage = (score > testResults.Length / 2) ? "Pass, Yay!!. Head to the 2nd
Floor" : "Fail, try again";
        testResultsText.text += "\nFinal Result: " + passFailMessage;
        Debug.Log("Pass/Fail message set");

        ResultsPanel.SetActive(true);
        Debug.Log("ResultsPanel activated");
    }

    // ========================= Desktop UI Buttons =====================
    public void PhisingIconClicked()
    {
        previousPanel = PhisingMenu;
        PhisingIcon.SetActive(false);
        PhisingMenu.SetActive(true);
    }
```

```
public void xButtonClicked()
{
    if (currentPanel != null)
    {
        currentPanel.SetActive(false); // Close the current panel
        rickRollPanel.SetActive(false);// Close the rickRoll Panel
        UNFloginPanel.SetActive(false);// Close the UNFlogin Panel

    }

    PhisingIcon.SetActive(true); // Show the Phising icon
    currentPanel = null; // Clear the current panel reference if needed
}

public void WifiButtonClicked()
{
    wifiMenu.SetActive(!wifiMenu.activeSelf);
}

public void StartButtonClicked()
{
    PhisingMenu.SetActive(false);
    PhisingTest1.SetActive(true);
    currentPanel = PhisingTest1;
}

// ========================= Rick Roll Link Button ======================
public void RickRollButtonClicked()
{
    rickRollPanel.SetActive(true);
    videoPlayer.Play();
}

public void CloseRickRollPanel()
{
    videoPlayer.Stop();
    rickRollPanel.SetActive(false);
}

// ========================= UNFlogin Link Button ======================
public void UNFLinkButtonClicked()
{
    PhisingTest2.SetActive(false);
    UNFloginPanel.SetActive(true);
```

```
    }

    public void CloseUNFloginPanel()
    {
        UNFloginPanel.SetActive(false);
        PhisingTest2.SetActive(true);
    }

    // ======================== Keyboard Buttons ====================
    public void BackButtonPressed()
    {
        PhisingMenu.SetActive(false);
        wifiMenu.SetActive(false);
        PhisingIcon.SetActive(true);
    }

    // Optional: Reset or retrieve score for other uses
    public int GetScore()
    {
        return score;
    }
}
```

Decryption Tool Script

```
using UnityEngine;
using TMPro;

public class CipherTerminalController : MonoBehaviour
{
    public TMP_InputField enterTextField;    // Input for encrypted text
    public TMP_InputField decryptedTextField; // Output for decrypted text

    private int currentLetterIndex = 0;  // Tracks the position of the current letter
    private char[] encryptedMessage;     // Array of characters representing the message being
entered

    void Start()
    {
        // Initialize encryptedMessage with a default length (e.g., 5 letters)
        encryptedMessage = new char[] { 'A', 'A', 'A', 'A', 'A' };  // or any desired length
        UpdateEncryptedMessageDisplay();
    }
```

```
// CipherUpButton: Move current letter one forward (e.g., A -> B)
public void OnCipherUpButtonPressed()
{
    UpdateEncryptedMessage(ChangeLetter(encryptedMessage[currentLetterIndex], 1));
}

// CipherDownButton: Move current letter one backward (e.g., B -> A)
public void OnCipherDownButtonPressed()
{
    UpdateEncryptedMessage(ChangeLetter(encryptedMessage[currentLetterIndex], -1));
}

// PushButtonNext: Move to the next letter in the text
public void OnPushButtonNextPressed()
{
    if (currentLetterIndex < encryptedMessage.Length - 1)
    {
        currentLetterIndex++;
        Debug.Log($"Moved to the next letter. CurrentLetterIndex: {currentLetterIndex}");
        UpdateEncryptedMessageDisplay();  // Update the visual marker
    }
    else
    {
        Debug.Log("Already at the last letter.");
    }
}

public void OnPushButtonPreviousPressed()
{
    if (currentLetterIndex > 0)
    {
        currentLetterIndex--;
        Debug.Log($"Moved to the previous letter. CurrentLetterIndex: {currentLetterIndex}");
        UpdateEncryptedMessageDisplay();  // Update the visual marker
    }
    else
    {
        Debug.Log("Already at the first letter.");
    }
}

// ExecuteLever: Runs the cipher decryption logic
public void OnExecuteLeverPressed()
```

```
{
    string encryptedMessageString = new string(encryptedMessage);
    Debug.Log($"Encrypted message entered: {encryptedMessageString}");

    // Example decryption: Caesar Cipher (shift by 3)
    string decryptedMessage = DecryptCaesarCipher(encryptedMessageString, 3);
    decryptedTextField.text = decryptedMessage;

    Debug.Log($"Decrypted message: {decryptedMessage}");
}

// Updates the encrypted message by changing the character at currentLetterIndex
private void UpdateEncryptedMessage(char newChar)
{
    encryptedMessage[currentLetterIndex] = newChar;
    UpdateEncryptedMessageDisplay();  // Show the updated message with a marker
}
private void UpdateEncryptedMessageDisplay()
{
    // Create a new string with the current letter underlined
    string markedMessage = "";
    for (int i = 0; i < encryptedMessage.Length; i++)
    {
        if (i == currentLetterIndex)
        {
            markedMessage += "<u>" + encryptedMessage[i] + "</u>"; // Underline current letter
        }
        else
        {
            markedMessage += encryptedMessage[i];
        }
    }
    // Update the input field to show the marked message
    enterTextField.text = markedMessage;
}
// Decrypts the message using a Caesar Cipher with a given shift
private string DecryptCaesarCipher(string message, int shift)
{
    string decryptedMessage = "";
    foreach (char c in message)
    {
        if (char.IsLetter(c))
        {
            char d = char.IsUpper(c) ? 'A' : 'a';
```

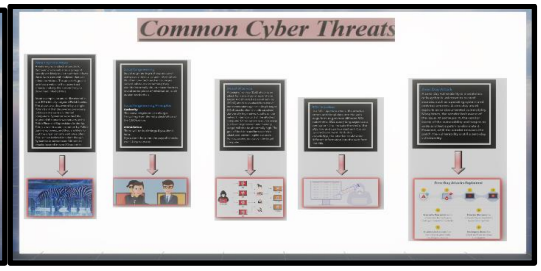
```
            decryptedMessage += (char)(((c - d - shift + 26) % 26) + d);
        }
        else
        {
            decryptedMessage += c;  // Non-letter characters remain unchanged
        }
    }
    return decryptedMessage;
}
// Change letter by a given shift (up/down)
private char ChangeLetter(char letter, int shift)
{
    if (char.IsLetter(letter))
    {
        char baseLetter = char.IsUpper(letter) ? 'A' : 'a';
        return (char)(((letter - baseLetter + shift + 26) % 26) + baseLetter);
    }
    return letter;  // If not a letter, return unchanged
}
}
```

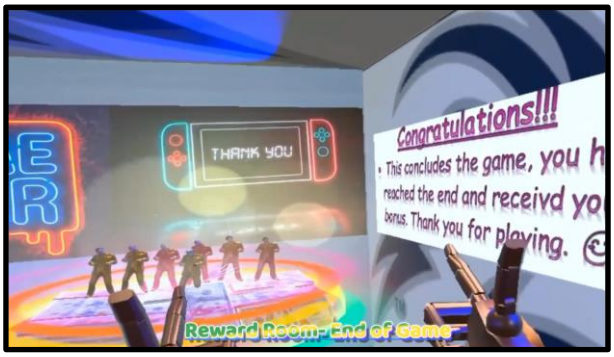
Mouse Cursor Script

```
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class UICursorController : MonoBehaviour
{
    public GameObject mouse;
    public float sensitivity;
    public XRBaseController controller; // Reference to your VR controller for input
    private XRDirectInteractor interactor;
    private XRBaseInteractable hoveredInteractable;
    void Start()
    {
        interactor = GetComponent<XRDirectInteractor>();
    }

    void Update()
    {
        // Update cursor position based on 3D mouse movement
        transform.localPosition = new Vector3(-mouse.transform.localPosition.x * sensitivity,
                                              -mouse.transform.localPosition.z * sensitivity,
                                              transform.localPosition.z);
```

```
        // Only trigger SelectEntered on click input
        if (hoveredInteractable != null && controller.selectInteractionState.activatedThisFrame)
        {
            interactor.StartManualInteraction(hoveredInteractable);
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        // Detect when cursor enters button area
        hoveredInteractable = other.GetComponent<XRBaseInteractable>();
    }
    private void OnTriggerExit(Collider other)
    {
        // Clear the interactable and end interaction on exit
        if (other.GetComponent<XRBaseInteractable>() == hoveredInteractable)
        {
            hoveredInteractable = null;
            interactor.EndManualInteraction();
        }
    }
}
```

Product Screenshots





Threat Analysis:

Door animation script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Once the key collides with the door, the door will open for the user
public class Dooranimation : MonoBehaviour
{
    public GameObject Door;
    public GameObject Secret_Key;

    // Collision between the key and the Door
    void OnCollisionEnter(Collision col){
      if (col.collider.name == Secret_Key.name){
        Door.transform.Rotate (0, -90, 0);
            Destroy(Secret_Key);
      }
    }
}
```

Changing computer screens script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Command_to_change_material : MonoBehaviour
{
    public GameObject First;
    public Image screen;
    public Material material;
    public GameObject Command;
    public GameObject Key;
    public Vector3 Position;

    // Once command interacts with computer, it will change screens and place the needed key
    // on table next to the computer
    void OnCollisionEnter(Collision col1)
    {
        if (col1.collider.name == Command.name)
        {
            screen.material = material;
```

```
            Key.transform.position = Position;
            Key.SetActive(true);
        }
    }
}
```

Hidden Door/Room Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Hidden_Door_contraption : MonoBehaviour
{
    public GameObject Wall;
    public GameObject Secret_Door_Sign;
    public GameObject Red_sign;

    // Secret door is set activewhen the user places the XXD command on the PC.
    void OnCollisionEnter(Collision col1)
    {
        if (col1.collider.name == "xxd command (correct)")
        {
            Secret_Door_Sign.SetActive(true);
            Red_sign.SetActive(true);
            Destroy(Wall);

        }
    }
}
```

Change Screen:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class change_screen2: MonoBehaviour
{
    public GameObject Button;
```

```
    public Image screen;
    public Material material;
    public GameObject NewButton;

    //changes from the screen to the area clicked on the screen.
   public void change() {
        Destroy(Button);
        screen.material = material;
        NewButton.SetActive(true);

        }
    }
```

Exiftool button:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class exiftool_screenbuttonaddon : MonoBehaviour
{
    public Image screen;
    public Material material;
    public GameObject NewButton;

    // Changes the virtual computer screen & initiates the new button for the user
    void OnCollisionEnter(Collision col1)
    {
        if (col1.collider.name == "Exiftool command(correct)")
        {
            screen.material = material;
            NewButton.SetActive(true);
        }
    }
}
```

Destroy:

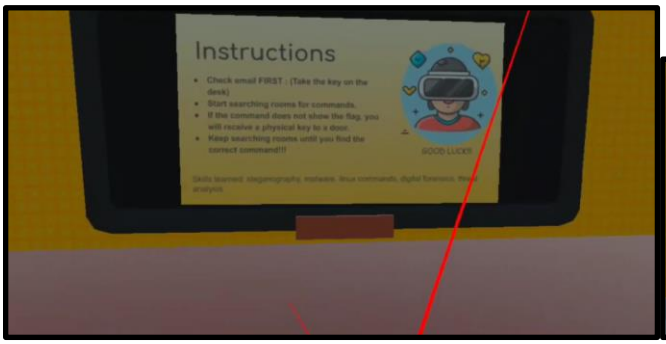
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destroy_screen : MonoBehaviour
{
    public GameObject Screen;
    public GameObject Button;
    // Used to destroy the Screen and Button items in the simulation
    public void Destroy_screen_object()
    {
        Destroy(Screen);
        Destroy(Button);
    }
}
```

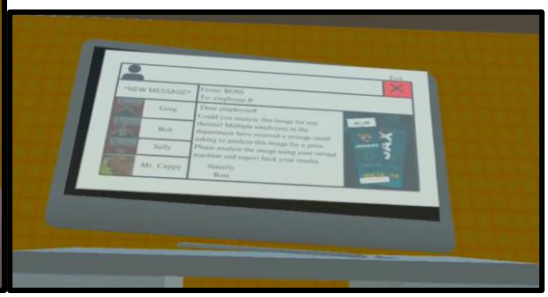
Change screen:

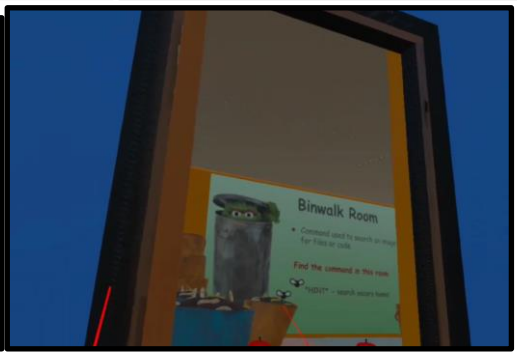
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

//Named as Command_to_change_2 since it is the second script used to change the screen
public class Command_to_change_2 : MonoBehaviour
{
    public GameObject First; //screen object
    public Image screen;
    public Material material;
    public GameObject Command;

    // Screen will change based on the correct command being placed on the PC
    void OnCollisionEnter(Collision col1)
    {
        if (col1.collider.name == Command.name)
        {
            screen.material = material;

        }
    }
```

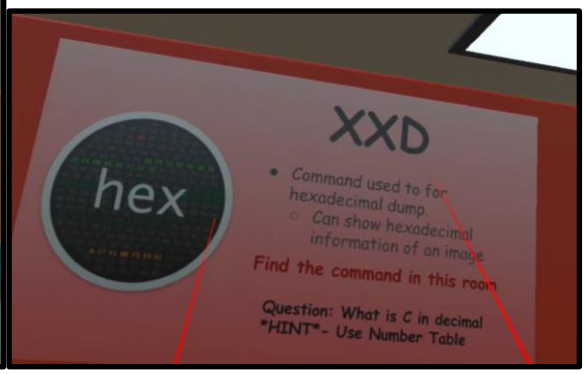
}

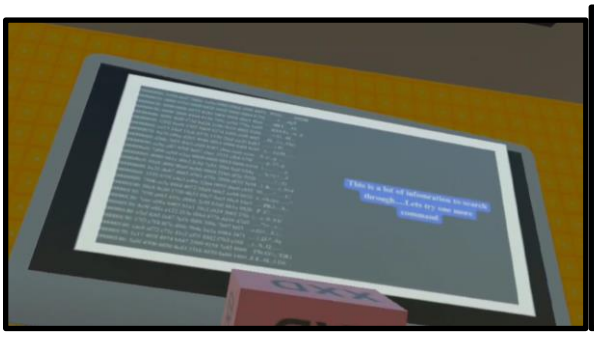
Product Screenshots

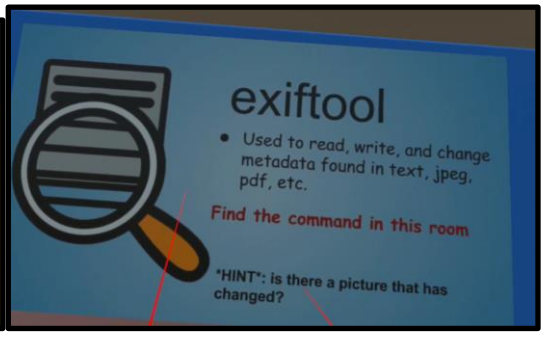


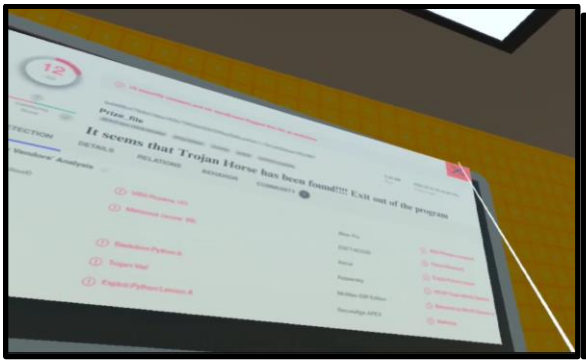


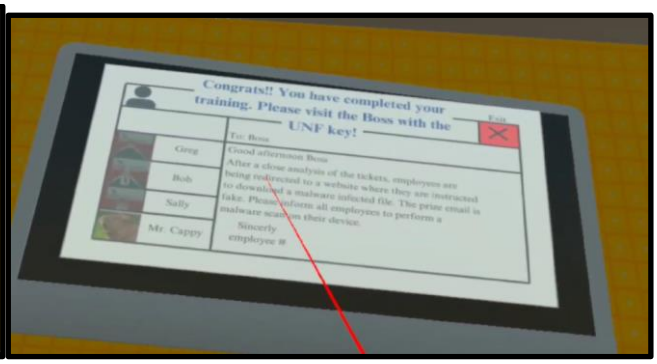


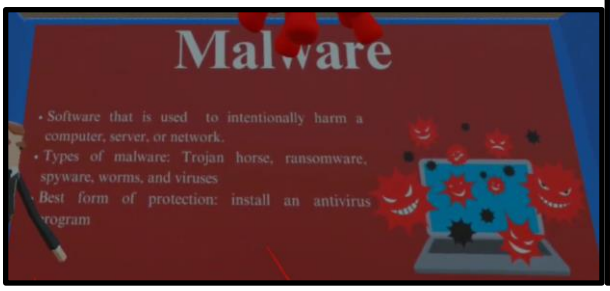


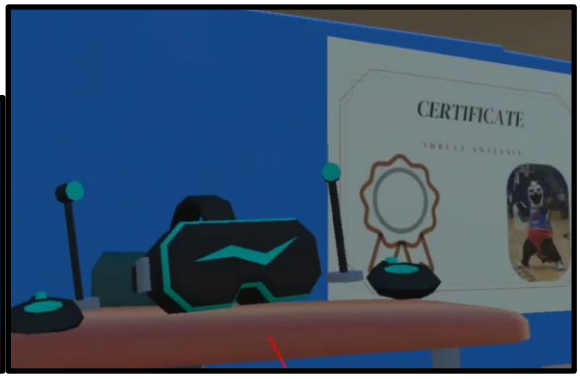












Incident Response:

Flashing Light:

using UnityEngine;

```
public class FlashingLight : MonoBehaviour
{
    public Light lightSource;  // Assign your light in the Inspector
    public float flashSpeed = 1f;  // Speed of flashing (higher = faster)

    private bool isFlashing = true;

    void Update()
    {
        if (isFlashing)
        {
            float intensity = Mathf.PingPong(Time.time * flashSpeed, 1f);
            lightSource.intensity = intensity;
        }
    }

    public void ToggleFlashing(bool flashing)
    {
        isFlashing = flashing;
        if (!flashing)
        {
            lightSource.intensity = 0f; // Turn off light when not flashing
        }
    }
}
```

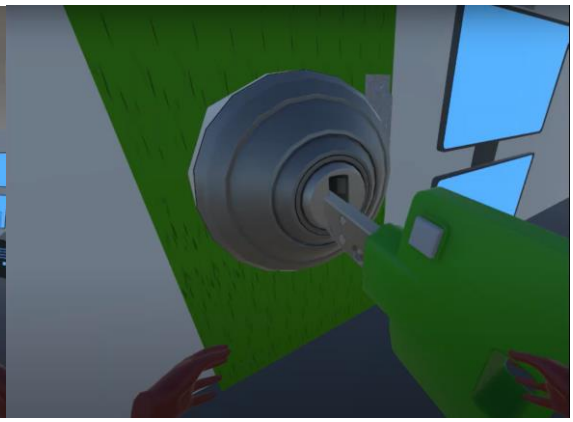
Mouse Controller Script:

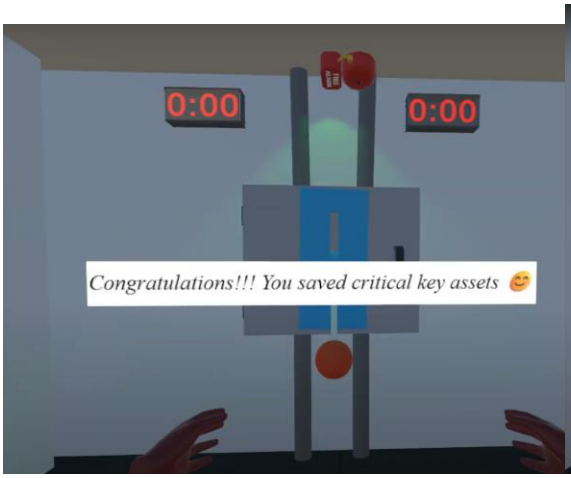
```
using UnityEngine;

public class MouseController : MonoBehaviour
{
    public Transform mouseCenter;
    public float maxXDistance = 0.5f;
    public float maxZDistance = 0.5f;

    // For managing the screens
    public GameObject[] screens;  // Array to hold all the screen GameObjects
    private int currentScreenIndex = 0;  // To track the currently active screen

    private void Start()
    {
        // Initialize: Set the first screen ("steering") to active
        SetActiveScreen(currentScreenIndex);
    }

    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("Hand"))
        {
            // Get position based on hand position
            Vector3 newPosition = new Vector3(other.transform.position.x, transform.position.y,
other.transform.position.z);
```

```
            // Clamp position to boundaries
            newPosition.x = Mathf.Clamp(newPosition.x, mouseCenter.position.x - maxXDistance,
mouseCenter.position.x + maxXDistance);
            newPosition.z = Mathf.Clamp(newPosition.z, mouseCenter.position.z - maxZDistance,
mouseCenter.position.z + maxZDistance);

            // Update mouse position
            transform.position = newPosition;
        }
    }

    // Method to move to the next screen
    public void OnNextButtonPressed()
    {
        // Deactivate the current screen
        screens[currentScreenIndex].SetActive(false);

        // Increment the index to move to the next screen
        currentScreenIndex = (currentScreenIndex + 1) % screens.Length;  // Loop back to the first
screen if at the end

        // Activate the next screen
        SetActiveScreen(currentScreenIndex);
    }
```

```
    private void SetActiveScreen(int index)
    {
        // Set the selected screen to active
        screens[index].SetActive(true);
    }
}
```

Product Screenshots:




0:00
0:00
Congratulations!!! You saved critical key assets


The data was lost, try again