Gamified Security Awareness For Developers Training Platform

Senior Design Team 7

Contributors

Iowa State University
Sri Charan Gurramkonda - Manager + Front-End Dev
Brayden Lamb - Design and Visual Lead
Caleb Lemmons - Backend-Dev + Game Scripts
Derek Lengemann - Front-End Dev + Testing
Charles Millar - Back-End Dev + Testing
Parker Schmitz - Technical Lead

Client

AllState Insurance Group
Suresh Kannan
Ethan Wilder

Advisor

Thomas Daniels

Dec 13, 2024

Executive Summary

Cybersecurity is a rapidly evolving field, and organizations are placing greater emphasis on raising awareness about critical principles, particularly those outlined by OWASP. While resources like HackTheBox, picoCTF, and CyberStart America exist to teach these concepts, they often lack the engagement, purpose, and storytelling necessary to effectively resonate with their target audiences, especially software developers.

To address this challenge, we developed CyEscape, a cyberpunk-inspired video game tailored for software developers at Allstate Insurance Group. The game blends immersive storytelling with interactive gameplay, teaching players about OWASP principles in a way that is both engaging and memorable. By incorporating creative elements and practical scenarios, it bridges the gap between theoretical knowledge and real-world application, making complex cybersecurity concepts more accessible and impactful.

This project was executed in partnership with Allstate Insurance Group, with key stakeholders: Sudesh Kannan, a Cyber Security & Privacy Professional, and Ethan Wilder, a Principal Security & Engineering Leader. Together, we worked to deliver an innovative solution that aligns with Allstate's mission to advance cybersecurity awareness and education across the tech field. Our goal is to provide a tool that not only educates but also inspires software developers to adopt and prioritize secure coding practices.

Learning Summary

<u>Development Standards & Practices Used</u>

- Utilized Agile software development methodologies to maintain flexibility and iterative progress, incorporating tools such as GitLab/GitHub for version control and project tracking.
- OWASP Guidelines for secure coding practices and to educate players and our primary audience of software developers at AllState Insurance Group on web application security.
- Engaged Unity Software for UI/UX development, ensuring interactive user experiences with high-quality graphics and versatile level design.
- IEEE 830 standards for Software Requirements Specifications to systematically define the software's functionalities, interactions, and requirements, ensuring project consistency.
- IEEE 29119 guidelines, which focus on software testing processes and requirements, providing a framework for planning, designing, executing, and evaluating testing efforts for software quality.
- IEEE 12207 standards for the software development life cycle, guiding the project through the key stages, while ensuring best practices for managing our game development.

Summary of Requirements

- Design engaging security challenges that simulate real-life cybersecurity threats.
- Embed practical coding exercises within the game to enforce secure coding principles.
- Character aesthetic consistency and ensure interactive capabilities within the game.
- Integrate OWASP Guidelines directly into the educational toolkit of the platform.
- Employ Agile-compatible project management tools for efficient development workflow.
- Facilitate internal and client communication using Microsoft Teams.
- Develop a cohesive visual theme and branding that supports the game narrative.
- Optimize the game for a wide range of operating systems and ensure it is resource-efficient.

Applicable Courses From Iowa State University Curriculum

- CYB E 230 (Cyber Security Fundamentals)
- CYB E 231 (Cyber Security Concepts & Tools)
- CYBE 331 (Cryptography)
- COM S 309 (Software Development Practices)
- COM S 227 (Introduction to Object-Oriented Programming)
- COM S 228 (Introduction to Data Structures)

New Skills Gained Outside Of Iowa State University Curriculum

- C# Scripting
- Unity Game Engine Software
- Game Sprite Development/Creation (KRITA Software)

Table of Contents

1. Introduction

1.1.Problem Statement

Our client, All State Insurance Group, presented us with a challenge: to develop an engaging game centered on the principles of OWASP for their new cybersecurity hires. For those unfamiliar, OWASP stands for Open Web Application Security Project, a comprehensive framework that guides the secure development of online applications, addressing the most critical OWASP Top 10 vulnerabilities.

In cybersecurity education, one of the challenges we've identified is the need for more practical learning opportunities as the field evolves. This is exemplified by the recent establishment of the Cyber Security Engineering major at Iowa State, highlighting this discipline's growing importance. However, the primary obstacle we aim to overcome is fostering engagement and retention within the tech industry. While there's no shortage of online coding challenges and practice platforms, few offer immersive experiences that connect with external companies, have a thrilling narrative, and feature in-game missions.

Our ambition with **CyEscape** is to shatter these conventions by crafting a narrative-rich adventure where every level is not just a challenge but a stage in a gripping story. Each stage is designed to introduce and explore a technical concept, progressively leading to a climactic showdown. This approach aims to enhance the learning experience by weaving technical skills into a captivating storyline. Initially, our plan was to create this game specifically for software developers. However, through ongoing conversations with our client, the scope evolved to target a broader audience, encompassing individuals with varying levels of expertise in cybersecurity. This shift allowed our narrative and concept to take shape in a way that aligned with our vision of creating an open-source Unity game accessible to all audiences!

1.2.Intended Users

Our product is designed to cater to the specific needs of three user personas, forming a cohesive part of our project's core mission. 1) Software Samuel 2) Human Resources Steve 3) Pro Gamer Lewis

First, there's **Software Samuel**, a developer striving to improve his secure coding techniques after discovering a security flaw in his website. Guided by the OWASP Top 10, our platform offers interactive coding exercises modeled on real-world security threats, providing Samuel with a practical and engaging learning experience. Next, we have **HR Steve**, tasked with recruiting top cybersecurity professionals for a company grappling with a talent shortage in this critical domain. Our platform supports Steve by offering tools to evaluate candidates' hands-on skills through practical challenges, progress tracking, and detailed feedback, enabling him to make informed hiring decisions. Finally, there's **Pro Gamer Lewis**, a former data breach victim determined to enhance his network and personal data security. Our solution empowers Lewis by providing effective strategies to protect his data from specific attacks featured in the game. In the game, we emphasize offensive security concepts, as these are central to the OWASP Top 10. By understanding these, Lewis can better prepare and ensure his system is rock solid against potential threats.

2. Requirements, Constraints, And Standards

2.1.1.Requirements

Functional Requirements

- Design security challenges replicating real-life threats to engage and test users in problem-solving. Incorporate practical coding exercises to apply secure coding techniques.
- Character aesthetic and movement capabilities should be consistent throughout game stages.
- Ensure that the game character can interact with the backgrounds on each level, such as issuing terminal commands through a computer, picking up objects, and jumping over walls.
- Implement a system for monitoring users' progress and saving current game progress.

Resource Requirements

- Directly integrate OWASP Guidelines into the platform as an educational toolkit. These guidelines offer a foundation for testing web application technical security controls and provide developers with a comprehensive list of requirements for secure development.

- Utilize Agile-compatible tools like GitLab or GitHub for efficient version control team-based development, and real-time project tracking.

- Microsoft Teams for communication with clients or among ourselves to discuss the game.

Aesthetic Requirements

- Develop a visually appealing, user-friendly interface to encourage continuous learning.

- Maintain a cohesive visual theme and branding aligned with the game narrative.

UI Requirements

- Employ Unity Software to develop the game, leveraging its features to create visually appealing and interactive user interfaces.

- Design various challenges using Unity's flexible environment, from basic (Social Engineering, Terminal Commands) to advanced threats (SQL Injections, DDoS Attacks).

- Ensure that the game character can interact effectively with Unity assets in-game. This includes engaging with environmental elements, manipulating objects, and executing actions.

- Leverage Unity's advanced graphics and assets capabilities to create a visually stunning game.

2.1.2.Constraints

- The platform must integrate with existing Agile development tools and OWASP resources without compromising their original functionalities.

- Ensure compatibility with diverse operating systems and devices, focusing on ease of access.

- The system should be optimized for low resource consumption, balancing high performance with minimal energy use. We don't want to overload anyone's computer.

- Visual design must adhere to predefined branding guidelines to maintain consistency and identity.

2.2.Engineering Standards

Comprehensive Documentation
Every phase of the game development process must be thoroughly documented to ensure transparency, facilitate clear communication among team members, and streamline the onboarding of new contributors. This includes but is not limited to design documents, development logs, code comments, and user guides.

Linking of Frontend and Backend
The frontend and backend components of the software must be designed and developed with a focus on seamless interoperability. This ensures the systems can communicate effectively, share data without loss or corruption, and function as a cohesive unit to deliver an optimal user experience. To achieve this, adhere to industry-standard protocols and interfaces and ensure thorough testing of integration points.

IEEE 830-1998 (Software Requirements Specifications)
This standard could be very useful as it outlines how to detail the software requirements effectively. It provides a systematic approach to capturing requirements that describe what the software will do, which is critical for both the development and educational objectives. Adhering to this standard will help specify functional requirements and the interactions between game characters and the environment.

IEEE 29119-2017 (Software Testing Standards)
This standard provides a comprehensive framework for testing software, with a focus on ensuring quality and security. In game development, we are constantly adding new features and stress-testing our levels after each addition to ensure an optimal user experience. This process involves system, interface, and regression testing. To guide us in achieving a structured approach to testing, we lean on IEEE 29119.

IEEE 1220-2017 (Software Lifecycle Processes)
The game itself is a software project, and IEEE 12207 provides best practices for software development, deployment, and maintenance. We utilized Agile as our primary software management framework, with the SDLC playing a critical role in tracking our progress through various development phases. This standard was important in identifying whether we were stalling in a phase or progressing too quickly, ensuring a balanced and efficient game development process. Furthermore, it proved invaluable for creating detailed reports and presentations to keep our client informed about the current state of the game.

3. Project Plan

Game Narrative

CyEscape is a cyberpunk-inspired video game where the narrative revolves around the player uncovering their identity piece by piece as the story unfolds. The game begins with the player waking up in a dark, dimly lit room, disoriented and unsure of how they got there or why they’re there. Nearby, a glowing monitor catches their attention, this serves as the first clue, signaling that they must complete challenges to uncover the mysteries of their surroundings. The setting is revealed to be Mirai Laboratories, and as the player progresses through these challenges, they begin to learn not only about the facility but also about their own identity and the events leading up to their transformation into a cyborg. The narrative is compelling and gripping, designed to immerse the player in a security-oriented storyline. We integrated various OWASP concepts into the gameplay, including challenges like injections, privilege escalation, weak passwords, and other real-world scenarios. These elements not only drive the story forward but also create an engaging learning experience for players as they unravel the truth behind Mirai Laboratories…

3.1.Project Management/Tracking Procedures

For developing CyEscape, we adopted the Agile Methodology. Our team's prior experience with Agile, both in academic and internship settings, provided us with the skills to manage the iterative nature of game development, ensuring continuous refinement and responsiveness to our client's feedback. We organized our work into 2-3 week sprints, complemented by weekly meetings every Tuesday from 1–3 PM. During these meetings, we replanned levels, developed and debugged features, and set clear weekly goals. These sessions also helped us better define our roles within the team, enabling us to complete three levels in the time it would typically take to finish two, thanks to improved resource management.

Additionally, we gathered feedback from our client every two weeks to ensure we were consistently aligned with their vision and expectations. This regular interaction allowed us to showcase our design progress and make necessary adjustments early. We were also fortunate to have Sebastian, a game designer at Allstate, assisting us with code debugging and helping us better understand Unity. His expertise significantly improved our workflow and contributed to the project's overall quality.

While we initially faced challenges, such as working without Git branches, we quickly adapted by adopting better development practices. For instance, we divided work based on specific parts of the game to avoid conflicts—one team member would focus on the environment while another worked on player mechanics. GitHub served as our central code repository, ensuring efficient version control, and we used tools like Teams and iMessage to manage tasks and communication. This combination of Agile practices, improved collaboration, and dedicated team effort allowed us to make good progress in the next semester.

3.2.Task Decomposition

Phase 1: Problem Definition & Research (Weeks 1-3)

- Completed a detailed report on our target audience, uncovering developer roles, challenges, and learning styles, which informed the direction of our game design.

- Identified many security threats, everything from social engineering to application security, setting a solid foundation for the game scenarios we would develop to address these challenges.

- Selected gaming elements and practical exercises for secure coding skills.

Phase 2: Platform & Game Design (Weeks 4-7)

- Developed a conceptual sketch of the platform, receiving approval based on UX expert review, which guided the subsequent design work.
- Created immersive scenarios that realistically simulate application security challenges, directly tied to the threats identified in Phase 1.
- Integrated OWASP cheat sheets into our game mechanics.

Phase 3: Prototype & Design Report (Weeks 8-12)

- We are developing a prototype for initial user testing. This phase is critical for iterating on our design based on client feedback, allowing us to refine our approach.
- A comprehensive design report is in the works, documenting our design choices, the integration of OWASP standards, and the rationale behind our game approach.

Phase 4: Prototype Refinement & Development (Weeks 1-5 of Semester 2)

- Our goal is to upgrade to a high-fidelity prototype, incorporating feedback from initial testing to enhance functionality and user experience.
- We plan to fully implement hands-on exercises within the platform, aligning with the educational objectives outlined in the project's conception.

Phase 5: Final Deliverables & Presentation (Weeks 6-10 of Semester 2)

- The project will culminate in launching a fully functional platform, aiming for at least 95% completion and 80% positive feedback from target audiences.
- A live demo and presentation will showcase the platform's features, engaging our audience and stakeholders with the final product—"CyEscape".

3.3.Project Proposed Milestones, Metrics, and Evaluation Criteria

Phase 1: Problem Definition & Research (Weeks 1-3)

- Milestone 1: Completing a comprehensive report detailing the target audience analysis, including developer roles, challenges, and preferred learning styles.

- Milestone 2: Identify and document the top five application-specific security threats.

- Milestone 3: Selection of immersive gaming elements and practical exercises for secure coding.

Phase 2: Platform & Game Design (Weeks 4-7)

- Milestone 4: The platform concept sketch development, including interface and navigation.

- Milestone 5: Creation of at least three immersive scenarios that simulate realistic application security challenges (ex. SQL Injections, Divide By Zero Vulnerability, Cryptography).

- Milestone 6: Integration of OWASP cheat sheets into the platform's design.

Phase 3: Prototype & Design Report (Weeks 8-12)

- Milestone 7: Development of a prototype for initial user testing.

- Milestone 8: Design report that includes the platform's design features and OWASP integration.

Phase 4: Prototype Refinement & Development (Weeks 1-5 of Semester 2)

- Milestone 9: Upgrade to a high-fidelity prototype with full functionality.

- Milestone 10: Implementation of hands-on exercises within the platform.

Phase 5: Final Deliverables & Presentation (Weeks 6-10 of Semester 2)

- Milestone 11: The application is 95% complete and receives an 80% or more positive feedback score from our audiences. Launch of the full-fledged platform, ready for evaluation.

- Milestone 12: Delivery of an engaging live demo and presentation.

Measuring Progress on Tasks

These tasks were measured through client feedback, internal reviews, and clear objectives for each milestone. While we couldn't gather precise percentages, we relied on word-of-mouth feedback from our client and group members. External user involvement was limited during development, but feedback received over Thanksgiving break and earlier helped us gauge progress. Our primary success marker was whether the game effectively conveyed the intended security concepts, whether we collectively felt proud of the levels we created, and whether it met our client's expectations. When feedback highlighted areas for improvement, we refined our design accordingly, ensuring the platform met stakeholder needs.

3.4.Expected Project Timeline

Sprint 1: Initial Setup and Design (Weeks 1-3)

- Brainstorm the initial game narrative and mechanics individually.
- Group discussion about starred ideas to build-up to an overarching concept.
- Sketch initial game designs and interfaces using Krita or Figma
- Discuss game elements with the client and request the assistance of someone who has experience building a game previously on an open-source game engine. This will bring us closer to identifying potential APIs, software, and other technical resources.

Sprint 1: Deliverables

- Initial game design sketches and prototype.
- Potential software applications for future usage.
- Choose the management strategy that works best for us (Agile).

Sprint 2: Unity Experiments & Refinement Of Levels (Weeks 4-6)

- Finalize numbers of levels and the first two to four levels should be designed.
- Implement feedback from clients and talk to our game designer for any directive on our levels.

Sprint 2: Deliverables

- Begin experimenting with our game engine Unity.

- Thinking about the Front-End & Back-End Interaction.
- Basic game mechanics (movement, terrain, other assets).
- Presenting our progress and experiments w/ the client.

Sprint 3: Finalizing Game Stages (Weeks 7-9)

- Finalize the game narrative with the client's feedback, rounding out all the specifics for the levels including setting, mechanics, play actions, etc.
- Have a well-defined scope for the project (number of levels in game, security challenges, etc.) because the following sprint will involve us designating technical tasks to each other.
- While exploring Unity, plan to prototype features that can later be implemented in our actual game. Additionally, consult with our game design resource at AllState for feedback on our game development approach to ensure it aligns with project goals and industry standards.

Sprint 3: Deliverables

- Final Game Documentation.
- Comprehensive Game Scope / Outline.
- Software Development Roadmap.
- Finalized Technical Resource List.

Sprint 4: Initial Development & Prototype Creation (Weeks 10-12)

- Begin development in Unity, focusing on setting up the basic framework, including the user interface (UI), game mechanics, and initial levels.
- Develop the first set of cybersecurity challenges and scenarios, starting with basic terminal commands and ensuring they are engaging.
- Implement a feedback mechanism within the game to collect initial user reactions and understanding, which is crucial for iterative improvement.

Sprint 4: Deliverables

- A working prototype featuring initial cybersecurity challenges.
- Internal playtest feedback report, highlighting areas for improvement.

Sprint 5: Feature Development & Content Expansion (Week 13-15)

- Based on feedback from Phase 4, refine existing game elements and expand the content to include more advanced OWASP cybersecurity scenarios.
- Integrate additional Unity features to enhance interactivity and user engagement, such as complex puzzles, scenario-based challenges, and in-game rewards for milestone achievements.
- Conduct further playtesting with a focus group to gather more targeted feedback on the game.

Sprint 5: Deliverables:

- Expanded game version with a broader range of challenges.
- Focused playtest feedback report, guiding further refinements.

3.4Updated Project Timeline

Goal	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Recap 491 + LVL1 - Player Movement - Setting The Scene - Sprite Creation - Clue Interaction - Beginner Cyber Security Puzzle W/ Terminal Cmd - Unity Basics	🟩	🟩	🟩												
LVL 2, 3 - NPC Interaction - Extending Scenes - Medium Difficulty Security Puzzles About Social Engineering				🟩	🟩	🟩									
LVL 4, 5 - Environmental							🟩	🟩	🟩						

Interaction W/ Terminal Cmd's - Networking - Looking For Weaknesses In A Sys/Network - Basics Of Security Encryption And Decryption W/ Labyrinth Puzzle - Fun 'Temple Run' Type Mini LVL - Minor Website Vuln. Hinting At Player;s Origins							🟩	🟩	🟩						
LVL 6, 7 - Focus On NIST Incident Response - Log Analysis - Attack Vectors - Shoot Projectiles Against Enemies - Prepare For Boss Level (7) Which Is A 'Choose Your Own Adventure' - Tie Together All Security Concepts Learned To Hack A Enemy Cyborg - Protect Player W/ Secure Coding Strategies (OWASP) - Lots Of Elements Go Into Boss LVL, So Ensure Fluidity In Security Ideas And NOT Overcommit										🟩	🟩	🟩			
LVL 7, Bug Fixes - Ensure Boss Level Is Functional And As Bug Free As Possible - Play Test Final Game And Make Any Technical Fixes - Saving Mechanisms - Upload To Unity Community Store													🟩	🟩	🟩

3.5.Risks And Risk Management

Phase 1 Risks	Phase 2 Risks
In the initial weeks of the project, we learned that one of our clients is from the United Kingdom, making our evening meetings potentially inconvenient for their attendance.	Choosing the wrong game engine could prevent us from implementing our game as envisioned.
A risk our client almost immediately pointed out is intellectual property. Will this game be ours, AllState's, Iowa State's, or something in-between.	Some resources required for the project may cost money. Does that come out of Iowa State's resource pool or the clients?
Mitigations	Mitigations
● Hold meetings in the morning. ● Come to agreement with clients to eventually make the game open-source, erasing the risk of intellectual property.	● Look into many different game engines. ● Keep the project as low cost as possible so neither party will be negatively affected.

Phase 3: Risks	Phase 4: Risks
Group members may not all agree on a level concept. It could be discouraging to throw out an idea and it is turned down by the rest of the group.	The character models available on the Unity asset store may not align with our envisioned characters.
If our ideas are overly ambitious, they might become unfeasible to implement later in the game's design process, leading to complications.	Outdated shared game code could lead to group members inadvertently working on tasks already completed.(Use GitLab/GitHub)
Mitigations	Mitigations
● Know that each suggestion can spark creativity and lead to better solutions. ● Make provisions for adjustments to finalized ideas, should the original concepts prove challenging to develop.	● Utilize the character models available to us, making modifications as necessary. ● Keep the group updated on progress or changes to the game's code to ensure everyone is informed and aligned.

Phase 5: Risks
There may be bugs in our game that could lead to undesired issues, requiring careful debugging and testing to ensure a smooth player experience. Additionally, our prototype may not be compatible with all operating systems, which could limit our audience or necessitate further development.
Mitigations
● Comprehensive documentation is crucial for troubleshooting and resolving bugs. ● Documentation can help users understand the systems with which our game is compatible.

3.6.Personal Effort Requirements

Tasks	Phase	Time (Hours)
1. Brainstorm Ideas for Game	1	10
2. Group Discussion Game Ideas	1	2
3. Sketch Initial Game Designs	1	4
4. Research Potential Software Applications	1	4
5. Develop Story of Game	2	50
6. Choose Software Application	2	1
7. Experiment with Unity	2	20
8. Finalize Scope of Game and Storyline	2	4
9. Develop Basic Framework of Game in Unity	3	20
10. Implement the first few basic challenges and levels	3	40
11. Play Test the Prototype	3	2
12. Use Feedback to Refine the Prototype	4	10
13. Design and Implement Next Advanced Levels	4	50
14. Play Test Advanced Levels	4	5

15. Have 3rd Party Users test Advanced Levels	4	5
16. Use Feedback to Improve Levels	4	5
17. Design and Implement Next Hard Levels	4	60
18. Play Test Hard Levels	4	10
19. Have 3rd Party Users test Hard Levels	5	50
20. Use Feedback to Improve Levels	5	10
21. Design and Implement Final Level	5	40
22. Play Test Final Level	5	10
23. Have 3rd Party Users test Final Level	5	5
24. Use Feedback to Improve Final Level	5	10
25. Test the Full Game	5	10
26. Release Game to Allstate to do an Alpha Test with Employees	5	1
27. Improve the Full Game	5	20

<u>Phase 1 Tasks</u>

- Conduct group discussion to share and evaluate game ideas, ultimately narrowing down the top concepts. Following this, we will hold an individual brainstorming session, allowing each member to further develop upon the selected final game ideas.

- Brayden sketched a few game designs so that we can give a visual along with our idea to the client to review and give the team feedback.

- The team was tasked with choosing a game development application for our project, which required conducting thorough research on various applications to determine the best fit.

<u>Phase 2 Tasks</u>

- We narrowed down our ideas to a final story concept, which we then presented to the client to gather feedback and ensure alignment with their vision.

- After a detailed meeting to discuss various game development applications, the team decided to use Unity for the development of our project, recognizing its robust features and flexibility.

- The team allocated a few weeks to experiment with Unity, allowing everyone to become familiar with its tools and functionalities to effectively contribute to the project.

- We held a meeting to finalize the scope of the project, deciding on the number of levels the game would include and outlining the storyline that would be followed across these levels. This helped set clear objectives and milestones for our development process.

Phase 3 Tasks

- The team will commence work on the game's core functionality, focusing on aspects such as character movement, user interface (UI), and interactions with the environment. This foundational work is crucial for building a smooth and engaging gameplay experience.

- We will design the starting level and establish basic challenges to test the game's functionality. This initial stage will allow us to assess the core mechanics, ensuring they align with our design.

- To ensure the game meets quality standards, the team will conduct internal playtests. This step is essential for identifying and resolving any issues before presenting the game to the client at the end of the semester and phase. This proactive approach helps us ensure a polished final product.

Phase 4 Tasks

- Use the feedback from showing the client our prototype to refine the prototype.

- The team will design and implement a more difficult and technically complex series of advanced challenges than those in the initial prototype levels.

- The team will internally playtest the game to find and fix those issues.

- The team will engage third-party users to test the game. These testers will provide valuable insights on the game's difficulty level and help identify any bugs.

- Utilize feedback from third-party users to refine and improve the advanced challenges.

- The team will develop and implement a new set of challenges that are even more difficult and technically demanding than the advanced levels, wrapping up the 6th or so stage.

- The team will either repeat the effective testing process used previously or enhance it with additional methods to ensure the game's quality and functionality further.

<u>Phase 5 Tasks</u>

- The team will design and implement the final few levels, introducing a variety of challenges that allow players to choose their path to one of several endings. These challenges in the final levels will vary in difficulty, ranging from medium to advanced, providing players with options.

- The team will conduct internal playtests of the full game to identify any issues with the complete product. Following this, we will engage third-party testers throughout the level creation to provide additional insights and feedback as per our usual process.

- We will present the completed game to our client for their review and feedback. Additionally, the client will have a select group of their employees playtest the game and provide their insights, ensuring that the final product meets their expectations and requirements effectively.

- The team will incorporate feedback from the client and their employee testers to refine the game further. After making the necessary adjustments, we will prepare the game for a public release.

3.7.Other Resource Requirements

Beyond financial resources, such as our Unity subscription, we also recognize the need for the expertise of a seasoned game designer. This professional can guide us through complex development stages and offer solutions when we face technical challenges that might delay our Agile sprints or leave critical bugs unresolved. Their expertise will be precious during planning sessions for future steps. Fortunately, we have access to such a resource through Allstate.

Additionally, engaging test users willing to interact with our game and provide detailed feedback on its mechanics and design is crucial. This feedback will enable us to refine and enhance the game iteratively. While our client will provide some of this critical input, the process and technologies involved in game development are somewhat familiar to us from our academic experiences. This familiarity allows us to operate independently and support one another effectively. Our ultimate goal is to foster a welcoming, problem-solving community that collaborates to deliver a high-quality gaming experience.

4. Project Design

4.1.Design Context

4.1.1.Broader Context

Public Health, Safety, And Welfare

Description: The project aims to enhance cybersecurity awareness and skills, focusing on safeguarding digital information. The OWASP Top 10 serves as an excellent framework for understanding and learning about common vulnerabilities. By equipping tech professionals with improved knowledge and abilities, the project contributes to strengthening the overall safety and security of the digital infrastructure.

Example: CyEscape introduces cybersecurity hires to real-world scenarios where they must identify and mitigate threats, mirroring situations they may encounter in their roles. This preparation helps prevent data breaches and cyber-attacks, directly impacting public safety by protecting sensitive information.

Global, Cultural, And Social

Description: CyEscape serves a global audience by providing education on universal cybersecurity principles through the OWASP framework. This approach addresses the needs of a culturally diverse workforce and promotes a more unified understanding of basic security practices.

Example: By integrating OWASP guidelines, which are recognized worldwide, CyEscape ensures that cybersecurity hires in any part of the globe are equipped with a consistent and comprehensive understanding of web security vulnerabilities. Additionally, we aim for our application to be publicly available once development is complete, allowing anyone interested to download it on Unity.

Environment

Description: While the project primarily focuses on cybersecurity, it also considers environmental impacts by promoting digital solutions that reduce the need for physical resources. The online nature of the platform decreases the carbon footprint associated with traditional, in-person training methods.

Example: By offering a digital learning platform, CyEscape reduces the reliance on physical educational materials and travel to training locations, contributing to lower energy consumption and less waste.

Economic

Description: Our game addresses economic concerns by enhancing the efficiency and efficacy of cybersecurity training, which is critical for businesses, especially those in the insurance and finance sectors, prone to cyber threats. Well-trained cybersecurity professionals can save organizations considerable costs related to cyber-attacks and data breaches.

Example: For All State Insurance Group, investing in CyEscape can result in long-term savings by minimizing the frequency and severity of security breaches. Specifically, CyEscape can be used to train new hires, equipping them with a solid foundational understanding of digital security principles and practices. This proactive approach to cybersecurity training enhances their ability to identify and respond to threats more efficiently, thus safeguarding sensitive data and reducing potential financial losses.

4.1.2.Prior Work/Solutions

Hack The Box

Hack The Box is a well-respected platform that offers competitive cybersecurity challenges. It serves a wide audience, including individual enthusiasts, corporate teams, and educational institutions. The platform provides diverse interactive, real-time challenges and scenarios designed to mimic various cybersecurity and penetration testing tasks. It has the steepest learning curve among the three applications, as evidenced by one of our group members who has experienced it firsthand.

PicoCTF

PicoCTF is a popular, ongoing online Capture The Flag (CTF) competition aimed primarily at high school and college students facilitated by Carnegie Mellon University. It provides a hands-on learning experience through challenges that simulate real-world cybersecurity issues. The platform is designed to introduce beginners to the field while providing enough depth to keep experienced individuals engaged, Additionally, PicoCTF occasionally hosts live hacking events, which allow participants to engage in real-time security competitions and further enhance their knowledge.

CyberStart America

CyberStart America is a dynamic online platform that hosts Capture The Flag (CTF) competitions, featuring realistic cybersecurity challenges similar to other platforms. However, its unique narrative-driven approach greatly enhances user engagement by seamlessly integrating storytelling elements into the challenges. This method makes the learning process more educational and entertaining, and it has served as our greatest inspiration for this project in terms of building levels/stages.

4.1.3.Technical Complexity

<u>Components/Subsystems:</u>

- Game Design and Narrative Development:

  Engineering Principles: Utilizes principles of software engineering and game design, including storyboarding, user interface design, and user experience optimization. The narrative-driven approach enhances engagement, requiring an understanding of cognitive psychology and learning theory to effectively intertwine educational content with engaging gameplay.

  Mathematical Principles: Physics of characters of movement, algorithms for random events, and logic for game progression and difficulty scaling per level.

- Security Challenge Integration:

  Security Principles: Based on cybersecurity principles and the OWASP framework. It involves understanding and applying advanced security concepts such as encryption, authentication, network security, and vulnerability assessment to create realistic and relevant challenges.

  Engineering Principles: The development of interactive, immersive simulations replicating real-world cybersecurity threats and defenses requires robust software development skills.

- Progress Tracking and Analytics System:

  Software Development Principles: Saving progress in the game involves using unique scripts for each level, designed to handle specific game states effectively. When a player leaves a level in-complete, such as Level 3, the game will not save certain elements—like doors remaining open or clues already picked up—thereby eliminating the need for additional coding to maintain these states upon the player's return. Critical elements contributing to challenge completion, such as an unlocked keypad, code saved on a terminal, or configurations within a virtual machine, are actively managed. This meticulous development process for the saving scripts is essential, as it ensures game continuity and user engagement without reverting any progress. Given their complexity and the precision required, these scripts are considered one of the most challenging aspects of our project, as highlighted by our game development resources.

- Cross-Platform Compatibility:

  Engineering Principles: Software is operable across various devices/operating systems, involving principles from computer science related to software portability, testing, and optimization.

Challenging Requirements And Comparison To Industry Standards

- Realistic Cybersecurity Simulation

  Challenge: The game simulates real-life cybersecurity threats and defenses more effectively than traditional training programs. It must stay updated with the latest security practices and threats, which require continuous research and development.

  Comparison: Our platform adheres to the standards of traditional CTF platforms by integrating real-time updates and feedback within a narrative context, a feature commonly found in established educational tools like Hack The Box or CyberStart America. We plan to draw inspiration from these platforms, as their challenges are well-refined and thoughtfully designed. This approach will significantly inform our own platform's development, allowing us to incorporate proven elements of success and expertise to enhance the overall user experience.

- 2D Sprite Game Format

  Challenge: Our project introduces a unique 2D Sprite game format to cybersecurity training, an area where few competitors currently operate. This innovative approach allows us to integrate gaming elements like those seen in "Voices of the Void", which utilizes in-game terminals to enhance player interaction and learning. We are positioned to create a highly engaging and educational experience by leveraging Unity's comprehensive game development resources.

  Comparison: Unlike traditional cybersecurity training solutions that rely heavily on text-based or video materials, such as CTFs or HackTheBox, our 2D sprite-based game introduces an engaging, interactive format designed to capture user interest and enhance learning outcomes. This approach represents a departure from conventional methods in the cybersecurity education industry, setting a new standard for teaching and understanding complex concepts through play.

- Accessibility and Usability

  Challenge: The platform must be universally accessible, adhering to WCAG guidelines for accessibility, and designed to be intuitive for users of varying technical expertise.

  Comparison: This initiative goes beyond typical cybersecurity training tools, which often do not prioritize accessibility or usability, thereby setting a new standard for inclusive design in technical training software. Additionally, our market analysis has shown that while competitors offer engaging challenges, they frequently fall short in user retention. Our objective is to balance a range of challenges while also ensuring that these challenges deeply engage users. We aim to replicate the compelling engagement that gamers experience, where they feel unable to stop playing, thus maintaining user interest and enhancing learning outcomes.

4.2.Design Exploration

4.2.1.Design Decisions

2D Pixel Sprite Design

A game can manifest in various forms—it might be 2D, 3D, or even exist within virtual reality. Games also encompass diverse genres, such as MOBAs and first-person shooters. Selecting the correct format is critical before advancing with any game development. Considering our time constraints, project criteria, and multiple consultations with our client, we have opted to create a 2D sprite-based game. This decision was influenced by the advice of a game developer resource experienced in both 3D and 2D game development who recommended the 2D route. Our vision for the game also aligns with inspirations from classic games like Flappy Bird and Pokémon. While 3D was initially considered, and platforms like Amazon Cloud Quest supported it, we recognized it as more complex and demanding a steeper learning curve than feasible within our time frame. Therefore, we settled on the 2D format, which has been the foundation of our brainstorming, idea generation, and game design efforts.

Unity Game Engine

We have decided to use the Unity Game Engine for our 2D game design. Although we also considered Unreal Engine and other platforms, Unity offers the best compatibility with our code management practices. Unity's support for 'assets', like sprites and interactive elements, simplifies game development.

Our consultations with our client and the detailed planning with our game developer resource were crucial in making this decision. He highlighted the advantages of Unity, including hosting capabilities, flexibility, and the overall development environment. To address collaboration challenges, we implemented Git branches to streamline our workflow and prevent conflicts. For example, one team member focused on developing environmental components like backgrounds, while another worked on player scripts, including interactions with surroundings and in-game challenges. This branching strategy allowed us to work simultaneously on different aspects of the game while avoiding conflicts in the codebase.

Level Creation (Choosing Challenges W/In Levels)

We knew the game would initially feature various levels, but the specific cybersecurity content for each stage needed clarification. Our original concept involved introducing cybersecurity concepts from the basics, beginning with minor security puzzles, social engineering and gradually progressing to more complex material. However, after discussions with our client and considering our primary audience, software developers, we opted to introduce terminal commands immediately and expand from there. This decision was crucial to the game's design, directly influencing the user's interaction and learning experience. Therefore, dedicating a few weeks (Sprint 1 & 2) to plan this aspect carefully was essential.

4.2.2.Ideation

Our most crucial design decision was selecting Unity as our game engine. Given the array of game engines available, each with its strengths and weaknesses, we thoroughly compared them to determine the best fit. We compiled a list of game engines, categorizing them based on their support for 2D, 3D, or both formats and providing a brief description and examples of popular games developed using each.

Unity and Unreal Engine stood out as the most popular among the contenders. While Unreal Engine is renowned for its advanced capabilities in creating top-rated games, particularly in 3D, it is also known for its steep learning curve. Conversely, Unity supports both 2D and 3D development, features a user-friendly interface, offers a wealth of features, and is celebrated for its realistic game physics. Other engines considered included GameMaker Studio, favored for 2D game development and its accessibility to both new and advanced developers; MonoGame, an open-source engine known for its OS support and its focus on 2D games; and PICO-8, a console ideal for beginners wanting to create small, retro-style games.

Despite the solid cases for other platforms, our decision to choose Unity was solidified by its extensive tools, resources, and tutorials, making it exceptionally versatile and accessible. Additionally, one of our trusted clients and a seasoned game developer, Sebastian, recommended Unity, validating our choice.

4.2.3.Decision-Making and Trade Off

Options	**Pros**	**Cons**	**X/10**
Unity Game Engine	● Publicly available assets ● Top-of-the-line game physics ● Supports 2D development ● User-friendly with tutorials	● Does not support more than one person working on a single project at a time	8
UnReal Game Engine	● Many top-hit games have been made using UnReal	● Heavily favors 3D design ● Steep learning curve	6.5
Gamemaker Studio	● Drag and Drop functionality ● Great for 2D games	● Ease of use limits capabilities ● Unfamiliar to clients	7
MonoGame	● Open-source (free) ● Games are compatible with many operating systems	● Uses Microsoft's XNA framework, which no one in the group is familiar with	6
PICO-8	● Simple and compact engine ● Creates retro style 2D games	● Simplicity limits functionality ● Great learning platform, but we are more interested in a high-quality product than a learning experience.	7

4.3.1.Overview

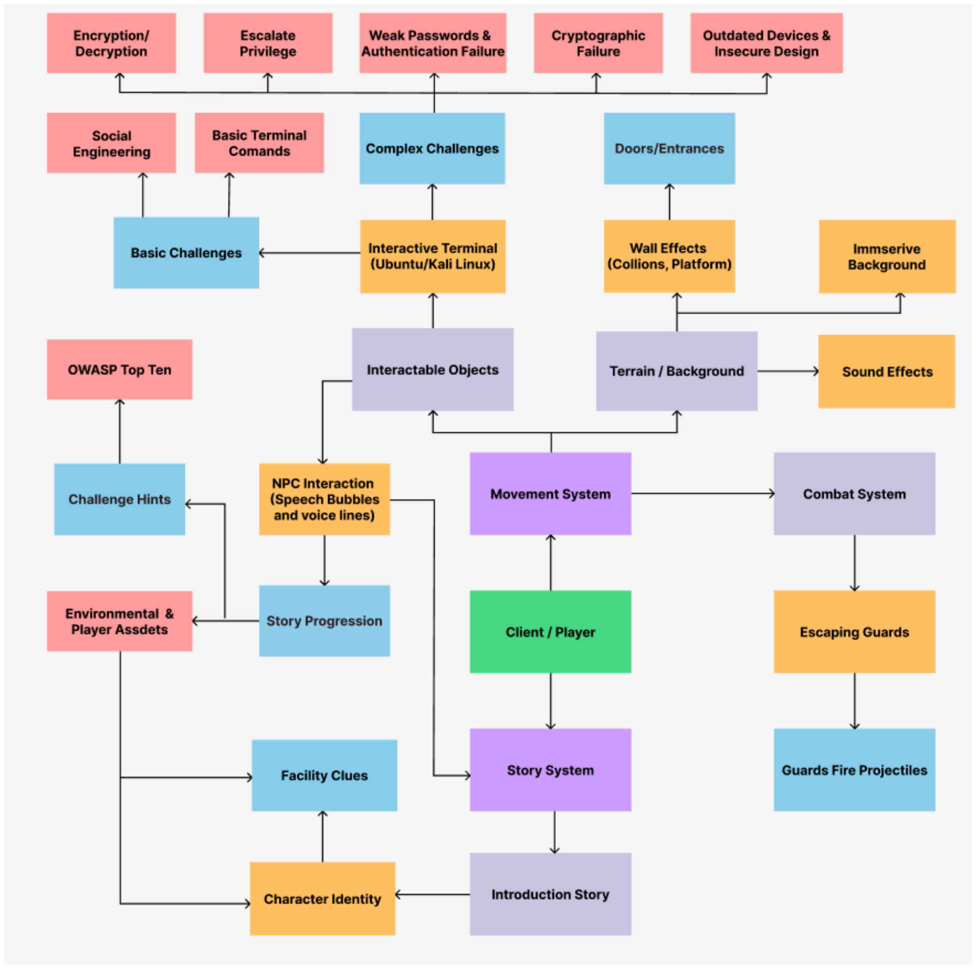
Our current project, CyEscape, is an engaging game that unfolds across 6 levels, each designed to enhance the player's understanding of cybersecurity principles and secure coding practices. It aims to educate users on crucial cybersecurity concepts, such as those outlined by OWASP, as well as basic vulnerabilities and exploits prevalent in the digital world. Key elements of our design are bulleted below:

- Unity Game Engine: This serves as the foundation of our game, enabling us to build and display the interactive environment where players embark on their security journey.
- GitLab: We utilize this platform as our codebase and version control.
- Team Collaboration: Perhaps the most vital component, our team's collaborative efforts and strategic planning underpin the entire project, transforming our vision into reality.

4.3.2.Detailed Design and Visual(s)

Our game has two central systems that control how it will function and how the user will interact with the game. The first system is Movement, which leads to Interactable Objects, Terrain/Background, and Combat System subsystems. The Interactable Objects system will involve NPCs that will give the player information and help them progress through the story of the game or will be able to provide hints and guidance to the user. Also, you can interact with terminals or machines so the user can learn and complete challenges to progress the story. For the immersive terrain subsystem, the user can interact with it (e.g., built-in terminal, NPC Interaction). The last subsystem of Movement will be the Combat System, where we will have 2 main types of combat. One type is Live Combat, where the user has to react to enemies in real time to pass the level. The second type will be Static Combat where the game will wait for the user's input to play out a scenario. The user won’t be able to react to the scenario until it is over.

The second central system is the Story System. First, we will introduce the game’s main character, and the user will play the point of view of the main character. At first, the user doesn’t know much about the main character and will have to progress through the game to find information about the main character. The user will get this information from NPCs or through files they see throughout the game. There will also be other ways of progressing the story like finding notes or reports in rooms. The end of the game will have different endings based on what path the player takes to escape the laboratory. Certain endings will be based on the method, but some will depend on items and information collected throughout the game.



4.3.3.Functionality

Our project is multifaceted; it features multiple levels where users engage with and solve cybersecurity puzzles. These challenges gradually increase in complexity throughout the game. To give you an idea, here's a walkthrough of what users will experience in the first level:

At the start, players are introduced to a 2D environment, navigating from left to right across the screen. The setting is a dimly lit room within the confines of "Mirai Laboratories," marked distinctly by its cold metal walls and the laboratory's emblem. The protagonist finds themselves disoriented, with no memory of their past, facing a door secured with a keypad that demands a six-digit code to unlock.

The game cleverly combines basic terminal commands with environmental storytelling. Pieces of information crucial for advancing the story are scattered throughout the room, revealing themselves as players interact with the computer terminal. Given that our primary audience consists of software developers, we've introduced the terminal without preliminary cybersecurity explanations, aiming to engage their technical proficiency directly. The initial level serves a dual purpose: familiarizing players with fundamental terminal commands (*ls, pwd, cat*) and establishing the game's interactive narrative.

As players progress beyond the initial level, the game delves into more technical challenges. The Unity game application's front end is the main interactive layer, while the back end processes all in-game actions and interactions. For example, when a player uses a terminal interface or launches a simulated cybersecurity attack on a website, corresponding scripts are executed within Unity to support these activities, ensuring a seamless and immersive gameplay experience. This structure enhances player engagement and deepens their understanding of security principles through application.

4.3.4.Areas of Concern and Development

The team and our clients agree that our current design plan meets the set requirements and addresses user needs. However, it is crucial to emphasize the term 'plan' since we have yet to begin implementing the video game. Our primary concerns are: 1) successfully creating a functioning video game and 2) effectively implementing gamified cybersecurity concepts. These concerns stem from our limited game development experience and how elements like mock terminals will be integrated.

To address these challenges, we are committed to thoroughly researching and familiarizing ourselves with Unity and C# to ensure that we can translate our design into a tangible game. We have also engaged with one of our clients, who is an experienced game developer, to gain deeper insights and guidance. He has provided valuable tips on managing different objects through controller scripts and effectively splitting tasks according to the game's level structure, clarifying our path forward. This structured approach will help us incrementally build our skills and confidence as we progress through game development.

4.4.Technology Considerations

In our project, we've opted for a mix of the Unity Game Engine, GitHub, and C# to develop our cyber security challenges within a 2D game setting. This choice provides my team with a solid foundation for game development, utilizing Unity's broad support for cross-platform development and its rich asset store, which significantly eases the creation and management of game elements. GitHub has been indispensable for us, offering a top-tier version control system that enhances collaboration by allowing my team to track changes and manage our code efficiently. We chose C# for scripting because of its compatibility with Unity, making integrating complex game logic and cyber security challenges straightforward.

However, we're aware of the limitations: Unity, despite its flexibility, can face performance hurdles with highly graphical or complex physics games and lacks comprehensive support for real-time collaborative editing, which has led us to assign team members to different game development aspects. Gitlab, while excellent for code management, demands a solid grasp of version control principles to prevent conflicts.

Choosing C# means my team needs to be comfortable with its syntax and paradigms, though its resemblance to Java, familiar from our coursework, aids with this learning curve.

4.5.Design Analysis

Our project is finally transitioning into the implementation phase after extensive planning. We've laid the groundwork with a detailed system design and are set to start development this weekend. Our approach for the Unity C# implementation is multi-faceted. We plan to employ GitLab for version control, creating separate projects for each game level to ensure organized integration into the final product. A significant focus will be developing object-specific controllers, including player controller scripts for managing story progression and generic helper scripts for essential functions like door operations. Additionally, we'll have model control scripts dedicated to centralizing player data. We've designated one team member to specialize in the game environment. In contrast, the rest of the team familiarizes themselves with the game's controls and mechanics, a crucial step before diving into development. Another aspect we need to consider is saving our project, which, according to Sebastian, necessitates a separate script entirely.

After having these client discussions and becoming more familiar with Unity, we organized the team into pairs, each tackling a specific game level, supported by two scriptwriters focusing on the game's scripting and logic. For example, with Derek, Charan, the project manager, will concentrate on front-end development for Level 1. Our game aesthetics lead, Brayden, will dedicate his efforts to creating the background, while Caleb focuses on scripting for future levels. Parker will work on Level 2, with Charlie also contributing there and delving into the backend. Regarding resources, as our client and our research proposed, we're leaning on the Brackeys YouTube Channel for development tips and TurnKey Linux for CMS solutions. This strategy outlines our current status and plans, highlighting the importance of thorough preparation and the potential challenges in building and testing our design.

4.5.1.Game Levels

Level 1: The Awakening

Overview:

The protagonist awakens in a metallic room, lying on a bed with no memory of how they got there or who they are. Disoriented and surrounded by cold, metallic walls, they notice a glowing terminal nearby. To escape, they must interact with the terminal, uncover clues about their surroundings, and discover a secret terminal command to unlock the only exit. This level serves as an introduction to basic commands, laying the foundation for the game's narrative and the protagonist's journey of self-discovery.

Key Objectives:

1. Exploring the Room:
    - Players can interact with objects in the room to find clues, such as a note with partially obscured digits of the passcode. You found a note: "Mirai Laboratories"
2. Using the Terminal:
    - The terminal serves as the player's first introduction to basic commands.
    - Commands:
        - help
          Output: Displays a list of available commands, including cat, ls, and exit.
        - ls
          Output: Reveals files like notes.txt and door_code.txt.
        - cat notes.txt
          Output: A cryptic hint: "Combine the pieces to unlock your way out."
3. Unlocking the Door:
    - Players must piece together clues from the terminal and the room to uncover the secret command needed to unlock the door. By exploring the terminal, they find a file named door_code.txt containing the command opendoor. Executing this command successfully unlocks the secured door, allowing the protagonist to proceed to the next stage.

Level 2: Hallway Encounter

Overview:

The protagonist ventures into the hallway, overhearing two employees discussing secret experiments. Following them, the player encounters a secured door requiring an access card. To proceed, they must use social engineering tactics, including disguising themselves as an employee, to gain further entry.

Key Objectives:

1. Eavesdropping on Employees:
    - The player overhears two employees discussing a specialist conducting a security audit.
2. Social Engineering:
    - After the employees leave, the player encounters a guard in the hallway who questions their presence. The player is given two speech bubble options: "I'm new here and forgot my access card" or "I'm from the IT department, just performing security updates." Depending on their choice, the guard follows up with additional questions, such as why the player forgot their ID card. The player must navigate these responses carefully, as failing at any point will result in the level restarting. Further builds narrative and the sequence emphasizes OWASP principles, particularly focusing on employee privileges.

Level 3: Discovering Forgotten Secrets

Overview:

The player uncovers secrets about their past by addressing OWASP Cryptographic Failures and Broken Access Control Vulnerabilities. They solve puzzles involving weak password hashes and escalate privileges to access sensitive data. A critical hint at the end leads to Level 4.The environment is a futuristic server room filled with glowing computers and data storage systems housing sensitive information. Interactive screens with command prompts, faint audio clues, and scattered hints.

Key Objectives:

1. File Exploration:

   Using the terminal, the player discovers files with restricted permissions (ls -l)
   ```
   -rw-r-----  1 root  root  4096  Oct 29  passwords.txt
   -rw-r-----  1 root  root  4096  Oct 29  experiments.pcap
   -rw-r-----  1 root  root  4096  Oct 29  web.net
   ```

2. Cracking the Hash:
   - Command: `cat passwords.txt`
     Output: A hashed password (8d969eef6ecad3c29a3a629280e686cff8ca6d1f…)
   - Command: `view-tools`
     Output: Displays tools like hashcat and a fictional crackzilla.
   - Command: `hashcat -m 0 -a 0 passwords.txt wordlist.txt`
     Output: The cracked password (cyborg123).
3. Privilege Escalation:
   - Using the cracked password, the player escalates to root access via `sudo -s`.
4. Unveiling the Logs:
   - Once root, the player accesses restricted files to uncover experiment logs (`cat experiments.txt`), realizing there is information on Experiment 23, his number.

Level 4: The Awakening of Experiment 23

Overview:

The player learns about their identity through terminal commands and audio logs, encountering OWASP Security Misconfigurations, Outdated Components, and Injection. The environment is a dimly lit research archive (e.g., science beakers, rusting novels, old desktop) with a terminal and scattered files.

Key Objectives:

1. Terminal Downgrade:

   - After failed login attempts, the player sees a prompt:
     "Terminal upgrade detected. Compatibility issue with commands. Consider reverting versions."
   - Command: sudo apt-get remove terminal_v2.0
     sudo apt-get install terminal_v1.5
2. Exploiting the Downgrade:
   - The player uses an SQL injection to bypass admin login: login: admin' OR '1'='1
   - Output: "Authentication successful. Welcome, Admin."
3. Adjusting Permissions:
   - The player alters restricted file permissions (chmod 750 experiments/) to gain access.
4. Audio Log Discovery:
   - Command: play experiments/audio_23.wav
     Output: Reveals the player's identity and suppressed memories.

Level 5: Regaining Control

Overview:

Captured and placed in a high-security prison, the player uses hacking skills to escape by exploiting OWASP Authentication Failures and Insecure Design of the environment. A dimly lit cell guarded by a single officer. Security cameras, interactive doors, and a terminal built into the player's interface.

Key Objectives:

1. Network Scanning:
   - Command: nmap -p- <IP-range>
     Output: Reveals the guard's IP and open SSH ports.
2. Exploiting SSH Access:
   - Command: ssh root@<Guard_IP>
   - Password: Default (toor).
   - Outcome: The player gains access to the guard's system.
3. Manipulating the Guard:
   - Command: unlock <cell door> to open the door.
   - Command: sleep <X seconds> to disable the guard temporarily.
4. Evading Detection:
   - An alarm is triggered, forcing the player to navigate the facility under pressure.

Level 6: The Final Escape

Overview:

The player must escape the facility after breaking out of the prison cell. This fast-paced level introduces projectile shooting mechanics as the protagonist fights off NPCs attempting to stop their escape. It's a high-intensity, action-packed finale that concludes with the player successfully escaping the facility, marking the end of the game. If the player could not escape in-time, the level will restart.

5. Testing

5.1.Unit Testing

Unit Tests

- Game Mechanics: Key elements of our game design include player movement, interactions with in-game objects like mock computers/terminals, as well as the behavior of NPCs and enemies.
- OWASP Concepts: Functions or modules responsible for implementing OWASP principles within the game, such as input validation, authentication, and secure communication protocols.
- Narrative Elements: Aspects of the game's narrative, such as dialogue systems, story progression, and character interactions, are integral components.
- Progression System: Features related to the game's progression, such as level unlocking, mission completion tracking, and player achievements. In addition to saving player progress.

Approach

- For each unit, we would write test cases to verify its behavior under various conditions.
- We will allow some 3rd party users to test the different systems and find issues.
- Tests would be automated to ensure consistency and repeatability.
- Test coverage would aim to cover critical functionality and edge cases to ensure robustness.

5.2.Interface Testing

Interfaces

- Unity GameObject Interfaces: Interaction between different GameObjects in the Unity scene, including player character, enemies, interactive objects, etc.

- Script Interfaces: Interaction between different scripts attached to GameObjects, such as player movement script, enemy AI script, interactive object script, etc.

- Unity UI Interfaces: Interaction between UI elements (buttons, panels, text) and game logic, including UI feedback for player actions and progression tracking.

Interface Testing Tools

- Unity provides its own testing framework called Unity Test Framework, which allows you to write and execute tests directly within the Unity Editor.

- Tests will be written in C# and executed either in the Unity Editor or an automated build pipeline.

- Use Unity's classes to develop scripts that can interact with GameObjects and in-game objects.

5.3.Integration Testing

Gameplay Mechanics Integration

- Essential for an engaging and immersive experience with fluid transitions between actions.
- Integration tests will simulate player interactions with various game mechanics (movement, combat, object interaction, puzzle-solving). Test cases will verify that the mechanics work together cohesively without causing unexpected behaviors or errors.
- Unity Test Framework can be used to write integration tests that simulate player inputs and verify the resulting behavior, say interacting with a terminal sprite.

OWASP Integration

- Since CyEscape aims to educate players about OWASP principles, the accurate integration of these principles into gameplay is crucial for achieving the learning objectives of the game.

- Integration tests will validate the implementation of security mechanics (input validation, authentication mechanisms, and secure communication protocols). Test cases will ensure that these principles are applied correctly and effectively within the context of the game.
- Unity Test Framework can be used alongside mock objects to simulate interactions with OWASP-related modules and verify their behavior.

Narrative and Gameplay Integration

- CyEscape employs a narrative-driven approach that integrates technical skills into a captivating storyline. This integration of narrative elements with gameplay mechanics is essential for immersing players in the game's world and significantly enhancing their engagement.
- Integration tests will examine the interaction between narrative elements (dialogue, story progression) and gameplay mechanics. Test cases will ensure that narrative events trigger appropriate gameplay responses and that gameplay progression aligns with the unfolding story.

Progression System Integration

- The progression system in CyEscape is crucial for maintaining player engagement, offering a sense of advancement and achievement. Its integration with gameplay mechanics ensures that player progress is accurately monitored and reflected within the game.
- Integration tests will examine how the progression system interacts with various gameplay elements, such as unlocking levels, tracking mission completions, and recognizing player achievements. These test cases will ensure that player actions effectively contribute to progression and that specific progression events initiate the correct gameplay responses.
- The Unity Test Framework can be utilized to develop tests that confirm the integration of the progression system with gameplay mechanics, ensuring they function seamlessly together.

5.4.System Testing

For system-level testing of CyEscape, our strategy involves a comprehensive approach that encompasses unit testing, interface testing, and integration testing. Unit tests focus on verifying the functionality of individual components, such as scripts and game objects, ensuring they adequately handle key game functionalities like player movement and enemy behavior. Interface tests check the seamless interaction between different game components, including gameplay mechanics, OWASP integration, narrative elements, and the progression system, to deliver the intended player experience. Integration tests validate the interaction and data flow among these components, confirming the proper integration of OWASP principles with gameplay and ensuring synchronization between narrative events and level progression.

5.5.Regression Testing

Through comprehensive regression testing, we ensure that new additions or changes do not disrupt existing functionality. This involves numerous automated tests that cover critical features such as player movement, enemy behavior, OWASP integration, narrative progression, and UI interactions. These requirement-driven tests are routinely executed using tools like the Unity Test Framework and are systematically documented for future reference. Moreover, we incorporate manual testing to complement the automated processes, providing an extra layer of assurance. By prioritizing regression testing, we plan to maintain reliability of our application throughout the Software Development Life Cycle.

5.6.Acceptance Testing

- <u>Client Involvement</u>
    - Stakeholder Reviews: Regularly schedule review sessions with clients to go through test results and product iterations. This involves demonstrating the functionality and discussing how it meets the listed requirements.
    - Feedback Incorporation: Allow stakeholders to provide feedback on each iteration of the game. Adjust testing and development priorities based on this feedback to ensure the product aligns with client expectations and user needs.
- <u>Functional Testing</u>
    - Scenario-Based Testing: Utilize scenarios that reflect real-world usage of the game by typical players to ensure all functional requirements, such as gameplay mechanics, narrative engagement, and progression systems, are validated.
    - Requirement Traceability: Map each test case back to specific requirements. This traceability ensures that all requirements documented in the project scope are tested.
- <u>Non-Functional Testing:</u>
    - Performance Testing: Test the game under various conditions to ensure it meets performance benchmarks such as load times, response times, and multiple users.

    - Usability Testing: Involve typical users in testing to evaluate the game's user interface and user experience, ensuring it is intuitive and accessible.

- <u>Security Compliance:</u>
    - OWASP Compliance Testing: Specifically test and demonstrate compliance with OWASP security principles to ensure that security requirements are met.
    - Data Integrity Testing: Ensure that data handling within the game, especially related to player progress and achievements, adheres to data protection standards.

- <u>Final Validation:</u>
    - Pre-Release Beta Testing: Conduct extensive beta testing with a closed group of end users to validate the entire game experience in a real-world environment. Collect and analyze data to make final adjustments before release.
    - Sign-Off: Prepare a detailed report summarizing the acceptance testing phase, including how each requirement has been met. Obtain formal approval and sign-off from the client.

5.7.Security Testing

Implementing all these security challenges into our game is beneficial, but it is crucial to ensure that vulnerabilities and exploits are contained within the platform and do not affect any remote devices. This is important as it directly impacts IEEE Standards and Digital Security Guidelines. We must train the platform to be self-contained and diligently cover all edge cases in our code to achieve a safe product.

5.8.Testing Results

Thorough testing was definitely our biggest obstacle. While we had multi-layered plans for it, we didn't manage to obtain concrete percentages or create diagrams to measure how successful or engaging the testing process was. Most of our initial challenges revolved around setting up the game environment, implementing player movement, integrating sprite sheets, and learning the basics of Unity. These tasks were more about understanding and becoming proficient with Unity rather than formal testing.

That said, the most frequently utilized testing methods were regression testing and system testing. Integration testing wasn't a major challenge, as Unity already provides a multi-OS 2D platform compatible with both PC and mobile setups. Through system testing, we ensured that each level consistently met our client's requirements. This was achieved through regular meetings, sprints, and iterative refinement of ideas, leading to improved levels. Regression testing became essential once we gained a solid grasp of Unity basics, as new elements were constantly being added to the game. These included creating and integrating terminal objects, NPCs, NPC interactions, projectile shooting mechanics, and more. Collaboration was facilitated through Git, and testing was conducted every Tuesday during our regular meeting times, ensuring consistent progress and addressing issues as they arose.

6. Implementation

Before we discuss the implementation needed for next semester, let us first reiterate what we aim to achieve by the end of this semester. By the end of this semester, we will have developed at least one or two mock levels with basic functioning, such as sprite movement, terminal interaction, and the basic process of developing backgrounds/terrain down to a T. It is simply about building off this process and iterating for the upcoming levels. To put it simply, take a look at our general process below:

1. Initial Planning and Visioning
   Conduct a group discussion to outline and visualize what Level X will entail. Record decisions and plans in the Sprint planner to track progress and ensure alignment with project timelines.

2. Asset Development
   Brayden, our design lead, will create assets, including clues, hints, and character designs relevant to Level X. Schedule a mid-Sprint review with the team to evaluate asset progress.

3. Front-End Development
   Concurrently with Brayden's asset design, the front-end team will start developing the mock-up for Level X. This includes researching implementations, analyzing the underlying code, and working on a basic implementation. Set regular touchpoints to sync with Brayden.

4. Script Development
   Develop any necessary scripts for the level, such as those required for game character interactions with commands like `pwd` or `ls`. Include these tasks in the Sprint planner and review their completion in Sprint meetings (Weekly On Thursdays).

5. Integration and Enhancement
   Once the basic framework is established, Brayden will integrate his assets with the front-end development, allowing further refinement and development of the level. Schedule a meeting with clients to gather feedback on the integrated version.

6. Iterative Development
   Repeat the development and scripting process as necessary for any additional scripts or assets. Use Sprint planning to allocate time for iterations based on feedback from the advisor and client.

7. Play Testing and Adjustment
   After the level development is complete, it undergoes play testing by each team member to ensure it aligns with the envisioned gameplay and interaction. Organize a client session to demonstrate the level and gather external feedback.

8. Save System Integration and Refinement
   The scriptwriter responsible for progress saving will then integrate relevant assets into the save system, with team support to refine and ensure the robustness of this critical functionality. Record this in the Sprint tasks and review in subsequent Sprint meetings.

9. Final Review and Level Integration:
   The level is considered complete once all elements are confirmed to work seamlessly and meet our standards. Schedule a final review with clients to confirm the completion and gather any last-minute feedback before integrating the level with the rest of our project on GitLab.

And so we will apply this general process to every level moving forward. Naturally, each level will have unique variations based on the specific challenges it presents, which vary in difficulty and narrative. Certain technical elements will remain consistent across all levels, such as accessing a terminal, navigating to remote sites, and securely coding in an emulator. Aesthetic elements like the game character's appearance will remain uniform to maintain a cohesive visual identity.

6.1.Design Analysis

Our design closely aligned with the model we envisioned. We aimed to create a 2D game with six levels, each incorporating distinct characteristics of the OWASP Top 10, and we succeeded in achieving a significant portion of that vision. One unexpected challenge was working with Unity to implement smooth and effective level transitions, which was difficult to perfect. Balancing a borderline realistic style within a 2D environment also proved to be tricky. Implementing the challenges was another hurdle, as most were terminal-based and focused heavily on command-line interactions. While OWASP typically addresses vulnerabilities like cross-site scripting and cross-site request forgery in web contexts, we adapted some principles into our game but could have expanded on this aspect further. That said, we managed to incorporate eight or nine of the OWASP Top 10 effectively. Our terminal functionality stood out as a strength, despite some redundancy across levels. The terminals supported features like tab completion, scrolling, command history, and executing challenges seamlessly. While the game was occasionally choppy and buggy as expected, these issues presented manageable challenges. Overall, we believe we delivered a strong result.

7. Professional Responsibility

7.1.Areas Of Responsibility

- <u>Work Competence:</u>
  IEEE places an emphasis on maintaining technical competence and undertaking tasks only if qualified. This aligns with the dedication to high quality work and integrity in the workplace. .

- <u>Financial Responsibility:</u>
  The IEEE Code stresses honesty in financial dealings and ensuring that services are performed to the best of their ability, which correlates with delivering services of value at reasonable costs.

- <u>Communication Honesty:</u>
  IEEE members are required to be honest and realistic in all professional evaluations, reports, and statements, which encompasses reporting work truthfully and without deception.

- <u>Health, Safety, Well-Being:</u>
  The Code calls for members to prioritize the safety, health, and welfare of the public in their professional work, which matches the goal of minimizing risks to safety, health, and well-being.

- <u>Property Ownership:</u>
  IEEE requires respect for intellectual property rights and making decisions consistent with the safety, health, and welfare of the public, including the protection of information.

- <u>Sustainability:</u>
  IEEE professionals are encouraged to improve their understanding of technology and its appropriate application to enhance the environment and communities across the globe, aligning with the responsibility to protect the environment and natural resources.

- <u>Social Responsibility:</u>
  The IEEE Code encourages members to seek, accept, and offer honest criticism of technical work and to be involved in public affairs. They are encouraged to produce services that benefit society.

The IEEE Code of Ethics generally shares similar principles with the NSPE Canon, but the IEEE might put a stronger emphasis on the role of technological advancement and the implications of engineering, such as sustainability and the global impact of technology. While both codes stress the importance of ethics in professional conduct, the IEEE Code may provide a more technology-centric view, reflecting the interests of its members who are primarily in the fields related to electrical or electronic engineering.

7.2.Project Specific Professional Responsibility Areas

- Public Safety, Health, and Welfare

  Applicability: High. The game educates software developers about cybersecurity, directly impacting the safety and security of digital projects they will work on, which affects public safety.

  Performance: High. Our project prioritizes ethical considerations and security principles which are critical to public safety in the digital realm. We also have IEEE Standards listed above.

- Service to Clients

  Applicability: High. Our team is dedicated to meeting the specific needs of AllState Insurance Group by developing a tailored educational game.

  Performance: High. Regular communication and tailored game development based on client feedback demonstrate strong service commitment.

- Public Disclosure

  Applicability: Medium. While the project itself might not require public disclosure, the educational content about cybersecurity could be beneficial for wider public awareness.

  Performance: Medium. If our team plans to share learnings or outcomes from the project publicly in any format, this could improve.

- Management of Professional Development and Competence

  Applicability: High. The project demands continuous learning and application of advanced cybersecurity and game development skills.

  Performance: High. Our team's use of Agile methodologies and regular updates indicates proactive management of skill and knowledge development.

- Fairness and Equity

  Applicability: High. Ensuring the game is accessible to users of different technical backgrounds and abilities is crucial. CyEscape has to be approachable and not impassionately daunting.

  Performance: High. Our commitment to inclusivity, such as adhering to WCAG and IEE guidelines/standards, shows a high level of performance in this area.

- Professional Accountability and Integrity

  Applicability: High. Ethical use of information, especially in cybersecurity, is essential.

  Performance: High. The team's adherence to engineering standards and ethical guidelines in developing cybersecurity training tools shows strong integrity.

- Sustainable Development

  Applicability: Low. The direct impact on environmental sustainability may be minimal, but the digital nature of the project promotes sustainability by reducing the need for physical materials.

  Performance: Low. While not a primary focus, the use of cloud-based solutions and energy-efficient practices contribute to sustainable code development (decrease CPU usage).

7.3.Most Applicable Professional Responsibility Area

The focus on developing a game that not only educates but also aligns precisely with our client's educational framework and requirements underscores 'Service to Clients' as the most crucial area of professional responsibility in our project. This ensures that the outcomes are not only technically proficient but also relevant and beneficial to both the client and the broader tech community, thereby strengthening our client relationship. Additionally, with the game set to become publicly available at the end of Fall 2024 semester and released to the general public, it significantly enhances this sector. This visibility means anyone interested in learning about security through an engaging platform might consider us a potential option. Therefore, we must strive to make their experience exceptional.

8. Closing Material

8.1.Discussion

Our project, CyEscape, has exceeded our client's expectations thus far. Our group has taken our Allstate client's rough idea of a cybersecurity video game and brought it to life. Our product is the game itself, and so far, we have thoroughly planned out the future development of CyEscape. At the beginning of the semester, this project was given to our group as a very open-ended idea. Our clients desired for our group to design a video game for them that will be used to teach developers about secure coding practices. There were no constraints on which game engine we should use, genre, or storyline. Over the spring semester, our group has planned out the entire game and pitched our ideas to our clients at Allstate. After pitching our game idea, we received positive feedback and constructive criticism on areas where our ideas may be challenging to implement. As we move into the development phase over the summer and the upcoming semester, we are committed to continuing to surpass our client's expectations.

8.2.Conclusion

The main goal of our project is to successfully plan and develop a cybersecurity-related video game to teach developers at Allstate secure coding practices, vulnerabilities, and exploits. We are on track to achieve this goal with our planning phase being complete and the development phase beginning. We have been following a plan to lead us to a successful final product. This plan includes thorough planning, presenting our work and ideas to our clients, meeting with game developers to pick their brains, and implementing our planned game design through the Unity game development platform. This plan will produce a quality product because our group has established a good relationship with our clients and game developers. These relationships will enable us to seek assistance when we encounter challenges within the project. While it's impossible to plan every aspect of a project with complete certainty, our strategy includes flexibility to accommodate unexpected circumstances and situations that may arise. Our aim to develop a cybersecurity-themed video game could be realized through various approaches, whether by crafting unique storylines, utilizing different game engines, or adapting existing games. Recognizing the potential of cybersecurity video games, our group aspires to ignite a trend that encourages the creation of diverse new games focused on cybersecurity education.

8.3.References

[1] "Unity real-time development platform: 3D, 2D, VR & AR engine," Unity, https://unity.com/ (accessed Apr. 16, 2024).

[2] "AWS Cloud Quest | Interactive Role-playing game | AWS," Amazon Web Services, https://aws.amazon.com/training/digital/aws-cloud-quest/ (accessed Apr. 17, 2024).

[3] "The innovative gamified Cybersecurity Learning Platform," CyberStart, https://cyberstart.com/ (accessed Apr. 16, 2024).

[4] US Cyber Challenge: Cyber Quests Spring 2024, https://uscc.cyberquests.org/ (accessed Apr. 16, 2024).

[5] "CMU cybersecurity competition," picoCTF, https://picoctf.org/ (accessed Apr. 16, 2024).

[6] Hack The Box, "About Us," Hack The Box. [Online]. Available: https://www.hackthebox.com/about-us. [Accessed: Apr. 16, 2024].

[6] CyberStart, "Home Page," CyberStart. [Online]. Available: https://cyberstart.com/. [Accessed: Apr. 16, 2024].

8.4.Appendices

Appendix 1 - Operation Manual

The game is intuitive, with simple controls for ease of use. The main movement keys are the left, right, and up arrow keys or the A, D, S, and W keys. To jump, press the spacebar. To interact with the terminal at any level, approach the desktop or computer and press the E key once or twice to have a pop-up.

Appendix 2 - Initial Version

Figma Projects

- Game Diagram
- Personas
- Project Management
- Ideation

Group Documents

- Gamified Awareness Doc
- Brainstorm Project Idea
- Brainstorm Mck Levels
- Final Level Scope
- Game Developer Notes

Appendix 4 - Code

9. Team

9.1.Team Members

1) Charan Gurramkonda	2) Caleb Lemmons
3) Charles Millar	4) Brayden Lamb
5) Derek Lengemann	6) Parker Schmitz

9.2.Required Skill Sets for Your Project

- Front-End Development - Developing of the background/terrain interactions the player experiences and how each level will function as per the specified challenge.

- Back-End Development - Scripting a realistic terminal, crafting challenge material, and implementing a robust system for saving game progress

- Sprite design - Designing an engaging background and characters for aesthetics of the game

- Game Design - Designing engaging cybersecurity challenges and an engaging story

9.3.Skill Sets Covered By The Team

- Front-End Development
  - Derek Lengemann
  - Charan Gurramkonda
  - Parker Schmitz

- Back-End Development
  - Charles Millar
  - Caleb Lemmons
  - Parker Schmitz

- Sprite & Background Design
  - Brayden Lamb

- Game Scripting
  - Charles Millar
  - Caleb Lemmons

9.4.Project Management Style

Our team employs an Agile management style, characterized by weekly meetings that enable collaboration, break down large tasks into smaller, more manageable components, and enhance accountability among team members. These regular gatherings also facilitate consistent feedback from our client, increasing our responsiveness and adaptability to any technical challenges that arise.

9.5.Initial Project Management Roles

1) Charan Gurramkonda - Front-End Development/Management

2) Caleb Lemmons - Back-End Development/Scripting

3) Charles Millar - Back-End Development/Testing

4) Brayden Lamb - Game Design/Visuals

5) Derek Lengemann - Front-End Development/Testing

6) Parker Schmitz - Front-End and Back-End Development

9.6.Group 7 Team Contract

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Face-to-Face Meetings. Thursdays 2:30 at the SIC.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. Google Drive for Files & Documentation. iMessage for general communication.

3. Decision-making policy (e.g., consensus, majority vote):
    a. Consensus

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Google Calendar will be applicable to track our team meetings. Charan will share the calendar with Team 07 and communicate if anything arises.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    - Each team member will prioritize and attend all weekly meetings, understanding the importance of these gatherings for our project's progress. In an unavoidable scheduling conflict, team members must inform the rest of the group as early as possible.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    - We rely on every team member to take ownership of their assigned tasks and complete them within the agreed-upon timelines. Recognizing that challenges may arise, team members must seek assistance or communicate difficulties early on proactively.

3. Expected level of communication with other team members:

- ○ We expect our members to respond to updates regularly. Given our schedules, we expect everyone to respond in at least 24 hours within notice.

4. Expected level of commitment to team decisions and tasks:
    - ○ Each team member is expected to fully engage with and commit to the tasks, contributing actively to our shared objectives. Importantly, team members are encouraged to voice their concerns constructively if they disagree with a decision or direction.

<u>Leadership</u>

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

    - ○ Charan: Team Organization + Client Interaction
    - ○ Brayden: Design lead for the Application
    - ○ Caleb: Information Application Lead
    - ○ Parker: Technical Lead
    - ○ Charlie: Testing Lead/Organizer
    - ○ Derek: Testing Lead/Organizer

2. Strategies for supporting and guiding the work of all team members:
    - ○ We plan to make the most of our weekly face-to-face meetings, not just as routine check-ins, but as opportunities to ensure that every team member is aligned with the project's objectives and progress. These meetings will serve as a platform for everyone to share updates, address challenges, and seek guidance. By doing so, we can collectively ensure that each member is on the right track and has the necessary support to succeed.

3. Strategies for recognizing the contributions of all team members:
    - ○ Our approach includes celebrating milestones and accomplishments, big or small. When a team member meets a deadline or completes their tasks efficiently, we will collectively acknowledge their effort with positive affirmations like "GREAT JOB!"

<u>Collaboration and Inclusion</u>

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

    <u>Charan Gurramkonda</u>
    Many of my classmates and I share a background in Cyber Security Engineering, but it's my internship experiences and extracurricular involvement that truly distinguish me. I've completed two internships at UnitedHealth Group: the first focused on file migration and working with the Microsoft ecosystem. At the same time, the more recent one involved a software development project using React.js, which will be highly relevant to our project. Additionally, I'm excited

about my upcoming cyber internship at Polaris, where I'll be joining the DN&IT Team, expecting to gain substantial knowledge. Beyond technical skills, my role as VP of Events for the Iowa State Engineering Student Council has honed my leadership and soft skills, equipping me to contribute effectively as a leader or a team player. With these diverse experiences and my commitment to excel, I am ready to give my all to this endeavor.

Brayden Lamb

I started as a Computer Engineering student but have transferred my major to Cyber Security Engineering. I have had engineering internships that weren't related to this type of project but showed me how to start a project and work with a diverse group of coworkers/teams. I was an officer and manager within the Gaming and Esports club here at Iowa State, which pushed me to keep on top of things and manage relations with other teams and colleges to fulfill my roles.

Charlie Millar

Like most of the group, I also majored in Cyber Security Engineering. In most large school projects up until this point, I have worked primarily on backend development. Over the past summer, I participated in research work at a tech company near my hometown. The research focused on the rapid advancement of AI and the new cyber security threats emerging from this advancement. The project's main focus was on social engineering. Outside of class and work, I am a member of the Theta Xi fraternity. I served as president of the organization for a year, and my term just ended with the fall semester. Being in this role has helped to develop my leadership and team skills.

Derek Lengemann

I am also majoring in Cyber Security Engineering. Not only have I done front-end work on multiple school projects, but I also have front-end experience working on multiple personal projects that I have worked on outside of school. As someone who has built multiple apps and games, my experience will not only help with the completion of the project but also with the teamwork side of the project. As a member of the game development club, I have experience working with a team and with and with developing various games. These experiences will help me with all aspects of this project.

Caleb Lemmons

Similar to many of my group mates, I am a Cyber Security Engineering Undergraduate with solid coding experience, mainly in Java, C, SQL, and Python. Much of my backend-oriented coding experience deals with database setup, management, and protection. I have also had prior internship experience working with Hexagon PPM as a Research & Development Intern under the head security engineer team. Here I was introduced to the expectations of Cyber Security engineers in the workforce and strengthened many of my technical skills and soft skills. I was lucky to be born in Saudi Arabia when my father worked for the Saudi Aramco Oil company. Through this experience, I met many people from different backgrounds and learned important communication skills at an early age, which has helped in situations like job interviews and group mate issues in team projects.

Parker Schmitz
I am a Software Engineering major with a lot of programming and software analysis experience. I am familiar with C and C-based languages such as Java, with a bit of experience in C++ and C#. I also occasionally do bash programming to automate certain tasks in Linux. I have worked on several programming projects in class, ranging from high-level Java programs to tinkering with an operating system. Outside of class I've worked on my own mod for a video game, and I run and configure my own game servers for my friends to play on.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

Charan Gurramkonda
Throughout my experience in various team environments, I've found that the most effective strategy for encouraging and supporting team contributions is through regular weekly discussions. My approach is to listen more and speak less, allowing me to provide decisive input when needed. I prioritize understanding and building upon my teammates' ideas. While I also propose my own ideas, I make an effort to weave them with the suggestions of others, ensuring that everyone's contributions are valued. This inclusive approach not only fosters collaboration but also enhances the collective output.

Brayden Lamb
I have been a part of many teams/groups and what makes a team succeed the most is that they meet/work together regularly and communicate issues. I am not someone who likes to take charge of a group but rather listens first to others' ideas so that I can work that into my view or idea to share with the group.

Charlie Millar
In a team dynamic, every team member can contribute in their own way. Everyone has different talents and skills, and it is important that each team member feels comfortable enough with the group to use their skills for the good of the project. I will acknowledge other team members for their work and make them feel appreciated as part of the team. Encouraging words are powerful and should be used to build the team up.

Derek Lengemann
As someone who has been a part of many group projects and teams, I believe that a successful team allows each member to contribute equally and provide input—utilizing the weekly meeting to ensure not only that each member is doing their part of the project but also for each member to provide input and to provide encouragement. The key to good teamwork is good communication; each team member can provide unique and valuable feedback and input.

Caleb Lemmons
Through past experiences with sports teams and school projects, encouraging and supporting contributions from each team member is vital for ensuring a collaborative and friendly environment. One practical practice is holding regular brainstorming sessions where members are invited to share their ideas and thoughts. It's essential to create a safe space where everyone feels comfortable speaking up, regardless of their experience level or background. Listening and showing appreciation for different perspectives also greatly encourage participation. Additionally, rotating leadership roles or responsibilities can allow each member to showcase their strengths and contribute in diverse ways. It's also beneficial to provide constructive feedback and recognize the efforts of all team members, as this can boost morale and promote a sense of belonging. Remember, a team that values and respects each individual's input is more likely to be innovative, productive, and successful.

Parker Schmitz
When discussing a part of the project, asking everyone individually for their opinion can be a good way to not only gain feedback, but to gauge everyone else's understanding of the topic at hand. Not only do we want everyone to be heard, and be a part of the group, we also want to ensure that everyone is on the same page and is not left behind.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment obstructs their opportunity or ability to contribute?)
    - Team 07 will use constructive criticism to build upon ideas and resolve conflicts. Suppose a team member has concerns about a particular idea or an issue with another group member. In that case, they are encouraged to express their thoughts openly in team discussions or in private conversations with a trusted team member.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

    - Adherence to Timeline: Our primary goal is to follow the timeline outlined in our proposal meticulously. This includes meeting all interim milestones and culminating in a live demonstration that meets the specified requirements.

    - Enhancing Communication Skills: We aim to improve communication skills through regular check-ins among team members and our advisor/client. This will ensure everyone is aligned and informed about the project's progress.
    - Deepening Cyber Security Knowledge: A key educational goal is to gain a more profound understanding of Cyber Security, explicitly focusing on secure application

development. We plan to achieve this through comprehensive teaching methods, including practical applications and theoretical learning.

2. Strategies for planning and assigning individual and teamwork:

    - Leveraging Individual Strengths: We will assess and understand each team member's unique strengths and skills to assign tasks effectively.

    - Clear Communication of Expectations: We will communicate expectations and responsibilities during our team meetings to ensure everyone understands their role.

    - Balanced Workload: We are committed to evenly distributing the workload among team members to prevent burnout and maintain high productivity.

3. Strategies for keeping on task:
    - To stay on track, we will prioritize effective communication, paying close attention to detail and deadlines. Team members' issues or struggles will be communicated promptly to seek assistance. Our weekly meetings will also serve as checkpoints to review that particular week's progress and address any impending tasks or concerns.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    - In the event of an infraction of our team's agreed-upon obligations, our initial approach will be understanding and constructive dialogue. A designated teammate will engage in a respectful and empathetic conversation with the member who has deviated from our guidelines. This discussion aims to identify any challenges that may have contributed.

2. What will your team do if the infractions continue?
    - If the issues persist despite our initial efforts and begin to affect the team's functioning and morale, we will escalate the matter for external intervention. This would involve bringing the situation to the attention of a TA or Professor.

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Caleb Lemmons	DATE 4/16/2024
2) Brayden Lamb	DATE 4/16/2024
3) Derek Lengemann	DATE 4/16/2024
4) Charles Millar	DATE 4/16/2024
5) Charan Gurramkonda	DATE 4/16/2024

6) Parker Schmitz DATE 4/16/2024