School of Informatics, Computing & Cyber Systems

Project Requirements

Serpent Studios

December 6, 2022

Sponsor:

Dr. Patrick Kelley

Mentor:

Italo Santos

Team Members:

David Hermann

Johnathan Ray

Tyler Morales

Nickolas Maxwell

Nick Shugrue

Accepted as baseline requirements for the project:

For the client:	For the team:
X____________________________	X____________________________
Date:	Date:

Table of Contents

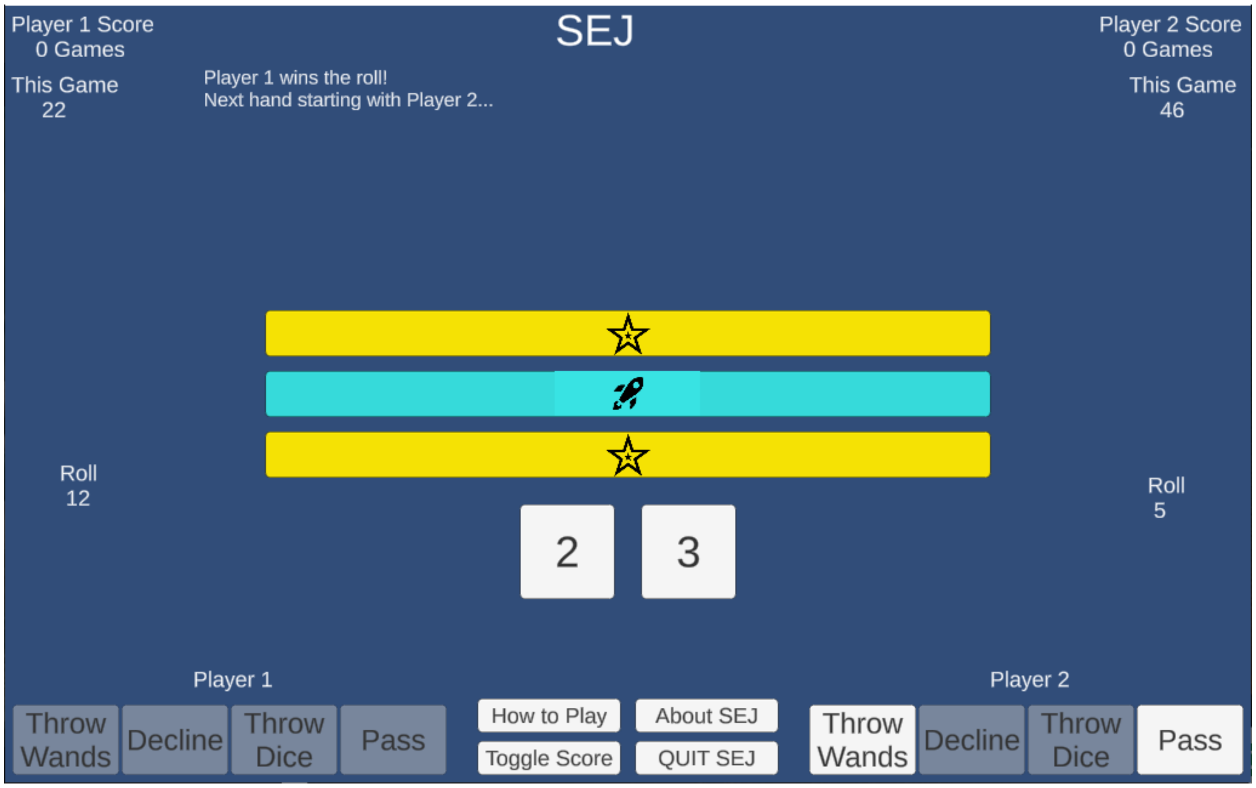
1 Introduction

Large companies such as Electronic Arts, Activision-Blizzard, and Bethesda with multi-million dollar franchises such as FIFA, Diablo, and Fallout respectively dominate the gaming industry. These studios produce games en masse, and as such ship products that are lackluster and stale. For example, a new FIFA game is released yearly but updates mostly consist of new characters based on roster changes made to teams registered with the International Federation of Association Football (FIFA). Recently, there has been an uptick in the gaming community's interest in original projects, new versions of older genres, or games that have been recreated from older works. A notable example is the roguelike genre renaissance which includes titles such as The Binding of Isaac, FTL: Faster Than Light, and Hades. This genre derives its name from Rogue, which was released in 1980. It features procedurally generated environments and permadeath, where losing a character means restarting a playthrough. New games often have innovative features either due to their rejection of conventional game loops or by improving upon previous design paradigms with new technology. Developments include engaging new stories which may be adapted from previous works of fiction that evoke nostalgia from older players while allowing a new generation of players to experience classic works in a fresh medium.

Large game studios inherit the resources of their parent companies whereas smaller independent developers cannot afford such tools. Previously, this would give larger studios an advantage. However, several smaller and more affordable "game engines", software packages with a large set of features for game development, have been made freely available to independent developers. Some larger studios have even adapted these game engines for their titles, demonstrating how powerful such engines are.

Our sponsor, Instructor Patrick Kelley, was fond of a game found in Serpent's Reach, a novel he read when he was younger. He wanted to know if the game was something that could be realistically played or simulated and set out to use his programming abilities to bring the game to life. Our client decided to use the game engine Unity and made a basic version of the game based on rules proved by C. J. Cherryh, the author. This prototype has simplistic two-dimensional visuals and an ambiguous user interface as to what is happening in the current game. Additionally, it does not contain features such as persistent scoring or peer-to-peer multiplayer. The prototype's code was written with only the developer in mind and as such it is inextensible;

no additions can be made without rewriting the entire codebase. While Sej has a scoring system for tallying points, it does not work in the prototype. As for multiplayer, the current implementation requires two players to share one machine and take turns inputting their turns in order, which leads to possible confusion and errors as to whose turn it is.



Player 1 wins the roll!
Next hand starting with Player 2...

Figure 1.1: A screenshot of Patrick Kelley's Sej prototype.

We have been tasked with fixing these problems and implementing new features that would allow for the game to support online multiplayer. This document contains the project requirements agreed upon by both the Serpent Studios team and Patrick Kelley. The Problem Statement section outlines the current implementation, shortcomings of the initial prototype, and other problems presented by the requirements outlined in the initial Sej Online proposal. Our proposed solution is outlined in the Solution Vision section. A list of detailed requirements is presented in the Project Requirements section, where requirements are grouped into functional, performance, and environmental requirements. The project plan and potential risks sections outline our work plan for the upcoming semester and possible risks that may develop. The conclusion section briefly summarizes the contents of this document.

2 Problem Statement

When our client created the initial game, his only goal was to see if it was possible to implement the rules of Sej found in Cherryh's book. While he was able to make a partially functional prototype, he was unable to flesh out many of the other aspects of the game. Some of the problems of the prototype include:

- The graphics are unintuitive and not appealing. The assets that our client used to represent Sej dice and wand game pieces are squares and rectangles with numbers and images that randomly change to simulate the player throwing them. A faithful, physical recreation of Sej would not use stationary objects that produce randomly generated values. It would utilize tangible dice and wands that provide immediate feedback to players.
- The UI is basic and has ambiguities in its design. It utilizes a default Unity Arial font and provides little to no information on whose turn it is or what actions the player can take to progress the game.
- The game also has minimal support for local multiplayer. Two players can play a game of Sej on the same machine, but the prototype does not have an implementation of online play nor does it have an artificial intelligence opponent for single player. These two factors prevent players from enjoying the game without another person.
- The prototype lacks a persistent scorekeeping system. Points are not stored from past games once the prototype program is closed.

3 Solution Vision

Our solution is an improved version of the version of Sej that our client created. This new version of the game will focus on four primary goals: implementing the Sej rules, a network-connected two-player mode, simple but attractive graphics/UI, and user accounts with cumulative scorekeeping. Figure 3.1 provides an overview of the basic structure of the game. In the following paragraphs, we provide a more detailed explanation of how our vision solves the problems found in the client's version of Sej.

- A graphical overhaul that properly simulates dice and wand rolls.
- An improved user interface that better communicates relevant information related to the player's score and actions.
- An online two-player mode that allows individuals to play with each other no matter how far apart they are.
- A way to keep track of how many games a player has won so that they can show others and keep track of how much they have improved.

To accomplish these goals, the team will use the Unity game engine. This engine is quite powerful and will serve as a multitude of tools for the project. The engine enables the use of plugins and other built-in tools which make things like multiplayer networking and creating user accounts much simpler to implement than if the team were to build these systems from the ground up. Additionally, Unity is capable of generating graphics that are clean and attractive, allowing players to quickly understand what is going on. Not only that, but the engine also provides libraries that are capable of producing UI elements that are customizable enough to allow simple and to-the-point interfaces that communicate to the player the necessary information they need in relation to the game.

The back-end aspects of this game consist of the online two-player mode and the creation of user accounts in a database. The online two-player mode will use Netcode, a networking library created by Unity that allows developers to create multiplayer games. The work that this library can perform on the back-end will allow players to search for another player they want to play with and join a server, all within a short period of time. The database will be implemented via short and simple scripts that can link the game and the database through the use of SQLite.

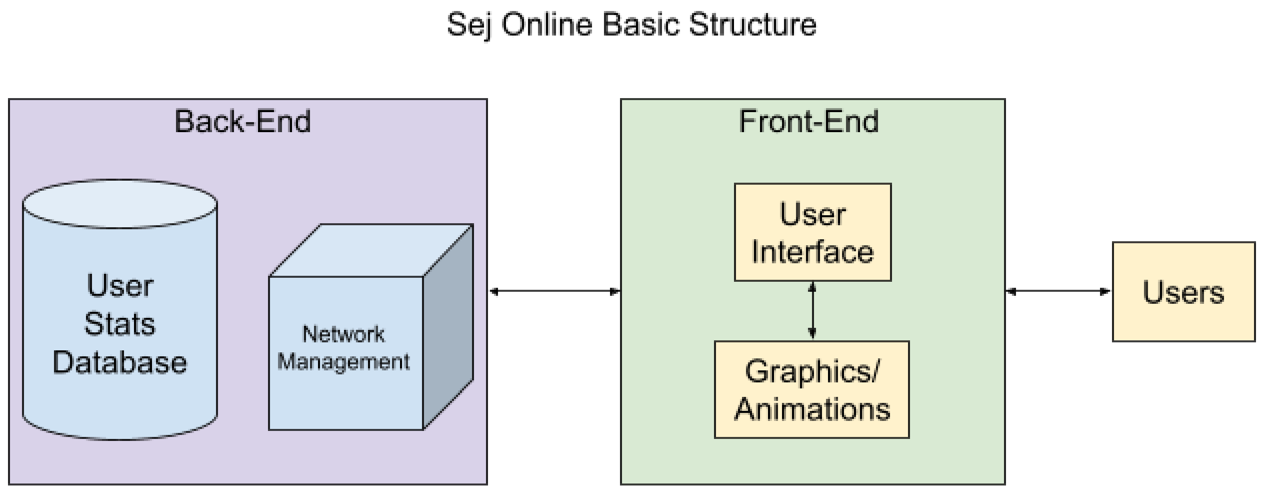


Figure 3.1: A simplified Sej Online structure.

4 Project Requirements

The following section will include an evaluation of the certain requirements that our product must fulfill. The requirements are split into three separate categories as follows:

- Functional Requirements - Specifications defining what our product must be able to do in order to function properly. In other words, what functions should be able to be performed for users of the product.
- Performance Requirements - Specifications of standards used to measure the capability of the product from the perspective of the user. In other words, how long it takes the user to complete some task while accessing one or more elements of the product.
- Environmental Requirements - Otherwise known as environmental constraints, these are essentially restrictions to the product related to hardware and software that must be used as decided by the client.

The above requirements have been identified through the process of requirements acquisition. The process included multiple meetings with the client and frequent discussion of the product amongst the members of the group. Documenting these requirements will be vital to implementation of the product in the near future. The specifics of these requirements will be detailed in the following subsections.

4.1 Functional Requirements

Sej is going to need numerous items. Local multiplayer connectivity, user accounts, user scores, an updated user interface, entirely new art assets, audio assets, and these are just the beginning. Building a game requires aspects from multiple avenues of software engineering. The multiplayer connectivity will need to have an architecture chosen, currently the choice is client-server. There will need to be tests to ensure the game is able to connect over various networks, and how the game performs on these different networks. The database will need to be secure enough that it will prevent players from accessing their own data and changing values such as points and/or wins/losses. The user interface will present challenges such as ensuring it is accessible for all players, and presents the relevant information in such a way that the game is easy and intuitive to play. Sej will require that the graphics are “pretty” to look at. They have to pop! Animations, sounds, all aspects of gaming need to be taken into account when creating the art.

The functional requirements for our product can be split up into several high-level categories, which can be found in the previous section. These categories represent the various components of our game that will be required to create a functional product according to the client. The categories mentioned are as follows:

- The rules of Sej must be implemented.
- There must be a network connected two-player mode.
- The graphics and UI should be simple and attractive.
- Users should be able to create accounts that have persistent score-keeping.

We will specify the functional requirements of each of the four high-level categories below.

4.1.1 Sej Rules Implemented

Our product is going to be a video game based on a gambling game from a Sci-Fi novel, so it will obviously need to implement the rules of that game as described by the author. The game is mainly used as a plot point in the novel, so there are some flaws to it, but the author was kind enough to include an appendix at the end of the book detailing how the game is meant to be played.

Sej is a two-player gambling game where each player takes turns rolling wands and then dice in order to score points. There are two six-sided dice and three four-sided wands. The sides of each wand have a unique face with a different meaning: black, blue with ship symbol, white, and orange with star symbol. Stars are worth twelve points and ships are worth ten; white and white with black equals five points for the pair of whites, but the black is played separately with its own value. White assumes the value of the highest color in the hand and only assumes black if both of the other wands are black. Black cancels all points in the possession of whichever player 'wins' the black wand. Play will always proceed starting with ships, then stars, and last black.

A hand begins by one player rolling the three wands. The players can then pass initiative to the other player or begin rolling the dice for possession of each wand in order to accumulate points. The first player to obtain one hundred points wins the game.

Use Case: Players take turns deciding how to play on the current hand

User: Player

Main Flow

1. Once a match begins the user is prompted to choose from a set of choices for play.
2. The user selects to roll the wands for the first hand.
3. After the wands are rolled the result is displayed and the user is given another choice.
4. The user chooses to either roll the dice on the current hand or pass initiative to their opponent.
5. If the user chooses to roll the dice, the system will then display the result and initiative will pass to the next user.
6. The next user now has to roll on the current hand to determine the winner of the wand in play.
7. The result is then displayed to both users, and the initiative returns to the first user.
8. This process continues on until either user accumulates a total of one hundred points thus concluding the match.

Alternative Flow

- At any point during the match either user may choose to forfeit thus concluding the match early. The other user is instantaneously victorious, then both users are disconnected from the match and returned to the menu.

Our functional requirements for this section intend on fulfilling the following game rules:

Basic Game Rules

A.1 Players will be able to roll wands at the start of a hand.

A.2 Players will be able to pass or roll the dice on any given hand.

A.2.1 Players will be able to pass initiative to their opponent.

A.2.2 Players will be able to roll the dice for possession of a wand.

A.3 Players will be able to decline a hand.

A.4 Players will be able to forfeit a match.

4.1.2 Simple but Attractive Graphics and UI

Graphics and animation are often considered one of the largest factors in keeping players interested. Art styles can be crucial to the feel that a player receives from a game, and some basic principles must be followed to ensure the best possible visual experience. Irrelevant animations may be distracting, and graphics that do not inform the player of what is occurring may leave them confused. High quality graphics and animations must be of paramount importance as the gaming industry is always pushing the latest in design. All visuals will be simple, as overwhelming the player often leads to dissatisfaction and low player retention. In the initial proposal, our client included a stretch goal for three-dimensional graphics. We will be implementing this by default to provide the best player experience. Besides this explicit requirement, no other items were discussed in the proposal. The team will discuss any additional changes with the client as necessary.

Using the Unity engine and the tools included, the team will be able to implement three-dimensional assets and simple but attractive animations. The game of Sej is difficult to pick up just by reading the rules alone, so anything that the graphics and UI can do to alleviate this learning curve should be done well. In short, there should be little ambiguity during gameplay. When a player's turn begins, they should be prompted to roll the wands. Once they do so, the face that the wands land on should be shown to the players. After this, each player should be prompted to either roll the dice or pass them to the other player. Once the round is over, the UI is updated with the newly calculated player scores. This cycle is repeated until the game is over. Throughout the game, the UI should display the option to forfeit the game is displayed to each player.

Use Case: The player is observing the actions occurring in a game and how they affect the information on the screen.

User: Players

<u>Main Flow</u>

1. A player is prompted on screen to roll wands when the hand begins.

2. Each roll is represented visually to players.
3. The option to pass or roll on dice is displayed to players for every hand.
4. After a round is completed, the user interface is updated with the scores of the two players.
5. The option to forfeit the match is displayed continuously to players on-screen.

Alternative Flow

- A player may select the option to forfeit at anytime during the match
- The players are shown a message displaying the victor of the match.
- Players are shown the main menu screen upon exit of the match.

4.1.3 Network Connected Two-Player Mode

One of the most significant goals laid out in the proposal is implementing multiplayer. The game needs to allow users to find another player that they want to play with and join a game without wasting too much time. To find someone to play with, the player will enter the username of whoever they want to play with. If the username does not exist, a message will state that that player could not be found and the player can try again. If the username does exist, the player can send a request to the found player to join a game, which they can either accept or deny. If they deny, the player who sent the request will be notified. If they accept, the two players will begin a game.

When they join a game, a server is created and one of the players will be assigned as the host. While they still serve the role of a client, their machine will also act as a server. Once the two players are in a game, they should be allowed to leave the game whenever they want and without causing any hassle on their opponent's end. For example, if both players finish a round, they should be offered the options of either playing another game or exiting the server. If they both decide to play, the server remains open and another game can commence. If both players decide to leave, the server should close automatically. If one player decides to leave, the other player should be notified that their opponent has left and that they should exit the server. This notification should also be shown in the event that one player leaves in the middle of a game, whether by their own volition or a sudden disconnect. The notification shown to a player may vary depending on who disconnects. For example, if the host disconnects for any reason, their opponent will have no choice but to leave as well. Because of this, the opponent's notification will tell them that they have automatically left the game. Additionally, if their opponent disconnects, the host will have to close the server themselves, so their notification will give them an option to close the server themselves.

Use Case: One player wants to play a game with another player online.
User: Player

Main Flow

1. The player who is looking for someone to play with (offering player) enters the username of whoever they want to play with and offers a game.
2. If the username exists, send an offer message to the player who has the matching username (receiving player), allowing them to accept or reject the offer.
3. If the receiving player accepts the offer, place the two players on a server.
4. Assign the role of "host" to the offering player.
5. Allow the players to play a game.
6. When the game is completed, ask both the players if they would like to play again.
7. If both players decide they want to play again, keep them on the server and repeat steps 5 and 6.

Alternative Flow

- If the entered username in step 1 does not exist, let the offering player know and ask them to input another username.
- If the receiving player rejects the offer from step 2, alert the offering player.
- If one player disconnects from the server in the middle of step 5, alert the other player of this and disconnect them from the server.
- If one or more players do not want to play another game in step 7, disconnect them from the server.

4.1.4 User Accounts with Cumulative Score-Keeping

Not very many games can function without some sort of scoring system to keep track of who is winning. In the current prototype of Sej, there is no persistent cumulative score keeping for users nor is there a way to keep track of users. When a user decides to play Sej Online, there must be some system in place to store a user's data. When a player faces off against an opponent the scores must also be stored somewhere as the game will need to access this data. Other user-tied data may include cumulative points, wins, losses, and previous match partners.

Sej will utilize an SQL database to store player data. When a user creates an account this information will be transferred to the database hosting all of the users Sej information. Any data relevant to the user that is generated during play will be stored in this database. This information is able to be accessed by the client to display various information to the user. User information including scores, wins, losses, and opponents. Using the Unity game engine, it is a rather straightforward process of writing a script that sends any of the relevant data to the database.

Scripts written in C# can be attached to game objects within the Sej world. These scripts will monitor for certain events, actions, or otherwise specified criteria involving their respective objects, and upon viewing such an event will write, read, or perform some other task with the database. The database can be easily implemented locally via SQLite as a plugin for the Unity game engine. This database will respond to changes from the game and can be stored locally for simplicity. There are further tools that can be used to secure these local files to prevent any malicious tampering. Security such as 256 bit AES encryption of the database files themselves.

Data can also be stored offline in a more secure location. By implementing a database using tools like MySQL and PHP, a more remote and secure database can be developed that interacts with similar scripts written for games developed with the unity game engine. Once a database is created and hosted, PHP scripts will act as intermediaries between the SQL database, and the C# commands coming from the Unity developed game.

Use Case:. Logging in

User: Players

Main Flow

1. A player enters login information (username and password).
2. Query is sent to the database checking if the account exists.
3. Upon confirmed response, a query is sent to check if login information is correct.
4. Player logs in.

Alternative Flow

- A player enters login information
- Query is sent to database checking if account exits
- Upon rejected response, invalid login attempt response is triggered

4.2 Performance (Non-functional) Requirements

The Serpent Studios team currently intends on using the Unity game engine to implement all of the MVP requirements as listed in the initial Sej Online proposal. For our networked multiplayer solution we aim to use a simple client-server multiplayer network architecture in order to implement Sej Onlines local multiplayer connectivity. This will be done through a series of scripts and or tools/plugins native to the unity game engine.Similarly, the database will be implemented similarly through a series of scripts that call out to a database hosted either locally, or on a remote server. These scripts will be capable of reading and writing data to the database.

4.2.1 Network Performance Requirements

As stated previously, the process of finding someone to play a game with and joining a server should be quick. Measuring the performance of this process requires testing the player search times and server connection speeds. When a player is searching for the name of someone they want to play with, the game will look through all the players that have an account with the game and return the specific player they want to play with. To test that this feature works properly, the team will measure the time between when the player searches for the username and when the game sends back a found or not found message. Once the username is found, we can then measure the time between when a request is sent to play a game and when that request is sent to the found player. Finally, we can measure how long it takes from when the found player accepts the request and when the two players join a lobby. Using this system we are able to accurately measure and verify the search times in order to maintain the most efficient online play.

Once the players are in a game, there are still several aspects whose performance we need to measure. For example, assigning the role of host to one of the players may result in one player having a better connection than the other. This delay should be as small as possible, and in order to measure it, we can put the two players' machines next to each other and measure the time between one player making an action and the other seeing it. These actions do not just have to be limited to those done in the game. The time it takes for one player to leave a game and another player receiving the notification indicating this not only need to also be measured, but should also be kept as small as possible.

4.2.2 Database Performance Requirements

When the client receives and sends data, much of it needs to be recorded. Player scores, wins, losses, opponents, usernames, and passwords, are all being used frequently by the game client. As the game uses these values they are frequently updated, and at this time the game client will utilize the scripts written for the Unity game engine to send this data to the database. If hosted locally, the data will be sent to a location within the device and encrypted. If hosted locally the data will be sent to a remote server, where a PHP script will intercept and manage any incoming packets, ensuring that the remote database is updated properly. Both methods are efficient, and secure.

When a player creates an account the game will prompt them to create a username and password. This information will be sent to the database and stored until needed. Upon future logins the user will be able to enter this previously created information in order to resume playing with the same points/wins/etc. Anytime players encounter each other this username data can be stored as opponent data for the opposing player, and vice versa, thus creating a unique persistent universe in which players may remember their competition. When points are gained or lost, a script will send this data to the relevant user account and store it in what can essentially be thought of as a table. Wins, and losses, any data that is needed may be stored here in this database.

4.2.3 Graphics and Animations Performance Requirements

The graphics and animations for Sej must be smooth, and visually appealing. Visuals that overwhelm tend to scare away players, rather the visuals for Sej should be inviting and encourage players to continue playing. All animations need to be smooth and free of jitters, it should be as if they were created by a professional studio. The animations should give the feeling that the game world has depth and is alive. Any sounds for the game must be created in such a way that they are simple and varied.

In choosing a basic set of qualities that will define our visual assets, we will ensure a verifiable and measurable system to gauge the effectiveness of the user interface. We will ensure that it implements simplicity, consistency, has purpose, strategy, typography, and proper communication in order to make it more intuitive and appealing. To do this, the UI should feel responsive and fast. Each interaction with the interface should feel natural and easy, with no room for lagging input or input that does not match that which the user has selected. Any buttons and sliders should be designed in such a way that they are unable to be used for any purpose other than their desired intent and are unable to be selected outside of their visual domains. The game should not use very much storage, nor should it need large amounts of memory. Ensuring that the game can be played on a variety of devices is crucial to its success, and as such the game should be designed with graphics such that it may be played on a multitude of devices, without sacrificing gameplay or speed.

4.3 Environmental Requirements

Our client has given the team the freedom to choose as they desire to bring Sej to life. In doing so, the team has found the freedoms quite enjoyable, and it was very simple to choose the initial tools to start up development. The biggest constraint that can apply to the actual development of the game is the Sej rules, which are to be implemented and followed with no modification. All MVP requirements such as persistent user scoring, local connectivity, and two-player competition will be implemented before any further features. Unity imposes various restrictions on the scope of projects. However, none of these constraints currently impact our project, or any future plans for development.

A potential constraint that we may run into is the legality of the game. At the moment, the team has no intention of developing the game as a commercial product. However, in the event that the game reaches a state that we believe could succeed commercially, we may not be able to sell the game due to the copyright of the source material. The team would have to contact the author and ask for permission before publishing the game. Again, however, there are currently no plans for selling the game, so this is not a constraint that we will be bound to in the near future.

5 Potential Risks

While there are several risks that come with this project, some are more severe than others. Figure 5.1 provides an overview of each of the risks discussed in this section, as well as the severity level of each of them. They can be divided into two sections: those that affect the developers and those that affect the player. The risks that affect development will be discussed first. As the team is learning and discovering new technologies, there may be changes along the way that makes these technologies no longer feasible as development continues on. This will lead to the team needing to implement a new course of action/and or a different approach to the relevant issue. Such issues in more detail could be situations where the unity game engine itself has errors when developing features for Sej; these errors may be hard to remedy without proper community/vendor support. Other problems include outdated tutorials, or tutorials that simply do not work anymore, and inefficient team training/ lack of training. While considering all of this, it is important to understand that Unity does a good job of keeping all of its tools up to date, so the likelihood of this happening is very low. That being said, the severity of this happening is high and could ruin some aspects of the game.

The risks that come with developing the game are important to consider, but it is equally important to consider those that the player can experience. The majority of them are more so an inconvenience than anything else (although there is one that will be touched upon momentarily). For example, the player does have the hardware necessary to play the game. Most of the team will develop the game on their personal computers that are not very high-end, so we do not intend to make a game that requires the best computer specifications. That being said, there may still be players that have outdated technology, such as computers that have operating systems from twenty years ago. They may have a modern operating system, but our game may not support it. Offering the game to different platforms can lessen the likelihood of this happening, but Unity does not allow developers to port their game to every platform ever made. Unity does allow games to be ported to WebGL, allowing for them to be played in a web browser. However, Netcode, the Unity library that supports multiplayer networking, does not support WebGL, resulting in players not being able to play online multiplayer at all. While the severity of this issue is of a medium level, the likelihood of this happening is relatively low due to the fact that Unity supports most platforms. There is also the risk that players may be incapable of online play if they live in an area with little to no connection to the Internet. This would limit their choices for how they want to play the game. The local multiplayer can alleviate this issue, but if there is no one to play with in-person, nothing can really be done. Both the severity and likelihood of this happening is medium. The lack of Internet is severe due to how dependent our game is on online play, and the likelihood of this happening is fairly high.

There is also a potential risk to players who have underlying health problems, specifically those with epilepsy. Many games use flashing lights as a way of providing feedback to the player. If a player fails an objective, lights could flash that indicate something negative happened. If a player completes a level, lights could flash indicating that they should be celebrating. These flashing lights can cause more harm than good, so we will strive to not put them in the game. That being said, this topic has only been discussed in the context of a developer putting them in their game intentionally. There are games that do not have them, yet they still create systems that allow them to happen spontaneously. For example, a game can have environments that use numerous colors and lights that do not flash. If the player turns the camera quickly, however, all those lights flash by in an instant and can cause the same effect as flashing lights. In short, the likelihood of this happening is at a medium level, and the severity of it occurring is very high. This is why, when developing our game, we must keep this risk in mind and ensure that this never or seldom happens.

Risk	**Severity**	**Likelihood**
Unity tools/assets become obsolete during development.	High	Low
Player does not have an Internet connection.	Medium	Medium
Player does not have hardware necessary to play the game.	Medium	Low
Game has flashing lights, intentional or not, that cause certain players to have seizures.	High	Medium

Figure 5.1: The risks that come with developing Sej Online, as well as the severity and likelihood of these risks.

6 Project Plan

At this point the team has set out five stages in the development process. The first stage is "Planning" and is the earliest stage in development. This stage should be considered the essentials and the "meat" of the CS476 course. Team development, and project initialization primarily takes place during this stage.

During our second stage of "Pre-Production" we begin the tech feasibility draft and move into the tech demo. In this stage, the team will complete the CS476 capstone course and all required materials. This stage is important as the team will finalize the development environment and all training required for the development of Sej. Over the course of this stage the team will begin to gather and create game assets, as well as design an outline for the initial MVP.

In our third stage of "Production", the team will begin working on the content for the MVP. Using all previously developed assets, and following the development route the team will implement features and test as this stage continues on. Starting with a basic game environment, the team will implement a simple and intuitive UI, followed by implementing the game rules as code. This will be followed by a period of testing to ensure game mechanics are working. Upon completion of game mechanic testing the team will move forward with development of the database, and networked multiplayer afterwards. This will complete stage three, and result in stage four "testing".

Stage four will ensure that multiplayer connectivity functions as intended.Various tests for disconnections, and other network related issues will be explored. The database will be monitored for any errors, and at this stage any security features will have been implemented. Upon the completion of testing, a final search for any and all bugs will be resolved in a hunt for completion.

The final stage is "Launch". During launch the team will build the product, and distribute it to all concerned parties. The team will participate in various presentations, and conferences at this time. Various demonstrations of Sej will be given during these events. Finally, after all has been completed, work on stretch features will begin. The team will begin to implement, adapt, and tweak the stretch goals with any time that is remaining to them.

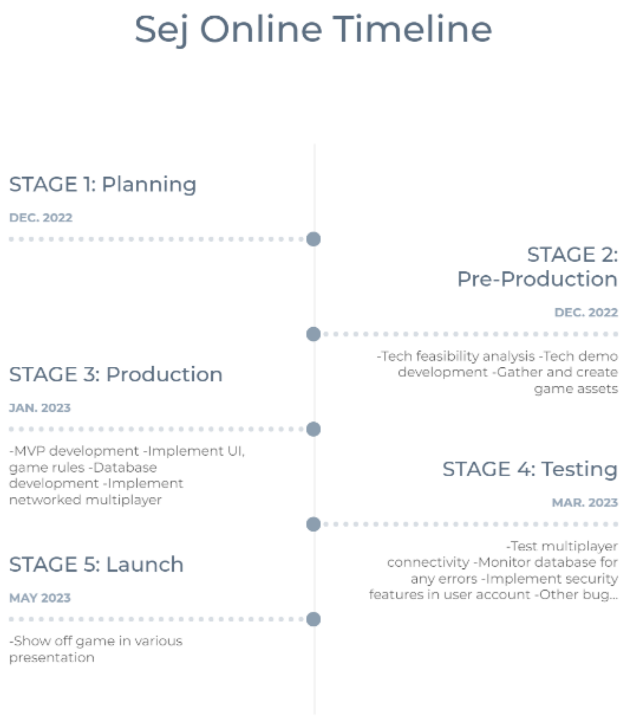

Sej Online Timeline
STAGE 1: Planning
DEC. 2022
STAGE 2: Pre-Production
DEC. 2022
-Tech feasibility analysis -Tech demo development -Gather and create game assets
STAGE 3: Production
JAN. 2023
-MVP development -Implement UI, game rules -Database development -Implement networked multiplayer
STAGE 4: Testing
MAR. 2023
-Test multiplayer connectivity -Monitor database for any errors -Implement security features in user account -Other bug...
STAGE 5: Launch
MAY 2023
-Show off game in various presentation

Figure 6.1: A timeline of the development process of Sej (December 2022-April 2023).

7 Conclusion

To conclude, the overarching goal of Serpent Studios while implementing Sej Online is to improve upon the prototype game made by Instructor Patrick Kelley, our sponsor. The main flaws with the prototype include a rough user interface, limited multiplayer, no persistent scorekeeping, and an unmaintainable codebase. Our goal is to implement revitalized versions of these features and ensure quality development while working on a fresh codebase. Currently, the project is nearing the end of the requirements phase as we have a written set of team standards, determined the project to be feasible, and done some preliminary research into possible technologies to solve aforementioned problems. While drafting this document, our team came across key insights such as how to handle lost internet connections and possible health implications of excessively flashy graphics. This document may not list out all requirements and problems we will come across in the coming semester but it is our hope that explicitly declaring the minimum viable product, some stretch goals, and foreseen problems will ensure a smoother development process than no planning at all. With a promising next semester on the horizon, we will deliver a game capable of both capturing new players unfamiliar with the source novel and pleasing older players familiar with a nostalgic story while challenging the standards for tabletop video games in an industry overshadowed by repetitive entries in tired franchises.

Glossaries and Appendices

Client - a computer that requests information related to the game from a central source, usually a server

Client-Server - a networking architecture that consists of clients and servers.

Host - a computer that acts as both a client and a server.

Server - a computer that provides information related to the game to one or more computers, usually the client(s).

Wand - a component of Sej that a player rolls at the beginning of a round and determines how many points are offered during that round.