SW Engineering Summer C 2023
[FAU: CEN 4010]

Battle Owls Game

Group 5
Team 🗿 (Moai)
Jesse McDonald (Scrum Master)
Nicklaus Becker (Possible Front End)
Alexander Ochoa (Possible Front End)
Noah Legault (Product Owner)
Andrew Donate (Possible Back End)

Milestone 3
7/18/2023
7/29/2023
Revision History

Executive Summary: Battle Owls (snake game)

Battle Owls is an engaging browser game that combines elements of the classic game Snake with multiplayer competition and customizable cosmetics. In this game, players control an owl character that navigates a vibrant and dynamic environment to consume worms, leading to growth and the accumulation of points.

The core gameplay revolves around guiding the owl to devour worms scattered throughout the game world. As the owl consumes more worms, it progressively increases in size, making it both more powerful and more challenging to maneuver. Players must skillfully navigate the owl, avoiding collisions with the owl's own body or any obstacles present, as such incidents result in the loss of points and potential elimination from multiplayer matches.

One of the standout features of Battle Owls is its multiplayer mode, where players can compete against others in a battle for supremacy. The objective is to amass the highest number of points by consuming worms within a specified time frame. The competitive nature of the multiplayer mode adds an exciting and intense dimension to the game, fostering an engaging and immersive experience for players.

To further enhance player engagement, Battle Owls offers a range of potential cosmetics. Winning multiplayer matches and leveling up rewards players with new outfits for their owls, allowing them to showcase their achievements and individuality. These cosmetics provide a sense of progression and personalization, motivating players to compete, improve, and stand out in the multiplayer arena.

Competitor Analysis:

Competitor Features	**Battle Owls Planned Features**
Classic snake gameplay with the single-player mode	Classic snake gameplay with single-player and multiplayer mode
A basic point system based on eating objects	A point system based on eating objects, with progressive rewards
No multiplayer mode	Multiplayer mode with “competitive” matches
Limited to no customization options	Extensive cosmetic customization options
Limited power-ups or unique abilities	Introduce power-ups and unique abilities for strategic gameplay
No level-based progression system	Level-based progression system with rewards

Planned Advantages/Competitive Relationship:

1. Multiplayer Mode: While some competitors lack a multiplayer mode, Battle Owls aims to attract a competitive player base by offering engaging multiplayer matches where players can compete for the highest points. This adds a new dimension to the gameplay and fosters a sense of competitiveness and interaction among players.
2. Extensive Cosmetic Customization: Unlike competitors with limited customization options, Battle Owls plans to provide players with a wide range of cosmetic choices, including outfits, accessories, and visual enhancements. This level of customization allows players to showcase their individuality and achievements, enhancing player engagement and satisfaction.

3. (TBD) Power-ups and Unique Abilities: Battle Owls plans to introduce power-ups and unique abilities that players can strategically use during gameplay. These additions will add depth to the game's mechanics, allowing players to employ different strategies and tactics to gain an advantage over their opponents.
4. Level-based Progression System: While some competitors lack a progression system, Battle Owls will implement a level-based progression system. Winning multiplayer matches and reaching new levels will reward players with new outfits and potentially other unlockable content. This progression system motivates players to continue playing and provides a sense of accomplishment as they unlock new rewards.

Overall, Battle Owls seeks to differentiate itself from competitors by offering a multiplayer mode, extensive cosmetic customization, power-ups and unique abilities, and a level-based progression system. These planned features aim to provide players with a more immersive, competitive, and rewarding gaming experience compared to existing offerings in the market.

Data Definition:

<u>Name:</u>	<u>Meaning:</u>	<u>Usage:</u>	<u>Comment:</u>
Owl Position	Pinpoint where the owl is on the grid	Shows where you are on the grid and make sure you have space to move	Able to see the position on the grid so you don't get confused
Owl Direction	Create vertical and horizontal movement for the player	User input to move the owl in certain directions	Helps users with input variables
Food Position	Randomly placed “food” that increases the owl's length	Food makes the owl grow larger for higher score	Random generated food across the grid
Grid	Two dimensional map that represents rows and columns	Shows exact location on the map by rows and columns.	Allows for use of arrays to track location easier.
Score	Players score which goes up the more food you get	Viewing points for competition.	Competitive edge to keep something to work for

Game Over	When you hit an obstacle or run into yourself and the game ends	This is when you lose and the game ends for hitting an obstacle or yourself	When you lose the game and have to start over
Speed	Control speed of the game updating and owl movement speed	Determines movement speed of the owl for the game	Different variations of speeds to increase difficulty
Dimensions	Length of width of the rows and columns	The dimensions of the map showing what you have to stay inside	Size of the grid
Owl Length	Current length of the owl	The longer the length the higher the score	Size of owl
High Score	Most food eaten and longest owl achieved	Used for creating a competitive environment to create a score to try and beat	Try and get the highest score to be the best
Level (TBD)			

Obstacles	Traps the if you run into will end the game	Inserted to try and mess up the player	Inserted obstacles to mess up the player
Power-Ups(TBD)			
Multiplayer (TBD)	Ability to play with other players online	2 or more players can play at once	Increase competitive nature by adding another player
Multiplayer Score (TBD)	Score of both players	Both players are able to see each others scores	
Lives(TBD)	If you die you will have another life to start where you left	Having another life if you die	Allows for one mistake for the player before the game ends.
Game Timer	Timer counting up or down to keep track of time in-game	Knowing how long you've played for	Helps the user to keep track of their time spent in the game.
Difficulty(TBD)	Different levels from easy-hard	Increases the hardness of the game	This will be increasing speed and other effects.

Overview, Scenarios, and use cases:

Overview:

One of the most well-known vintage games is The Owl Game. The player controls an Owl, who is the main character in the game. The Owl must be fed in order to grow longer while avoiding obstacles and its own wagging tail. Until the Owl reaches the game's edge or its own tail, the game is still in progress.

Scenario:

The playing field in the Owl game is a grid with a set size. At the beginning of the game, an Owl is placed at random in the grid. Additionally, in the beginning, a food item is dropped into a random spot on the grid.

The Owl begins to travel in a specific direction as the game starts. The Owl's movement is controlled by the player (up, down, left, or right), with the goal of eating the food. The length of the Owl and the player's score both grow as the food is consumed by the Owl. Once the food has been consumed, a random piece of food will next appear somewhere else in the grid.

The length of the Owl affects how difficult the game is to play. The game is over if the snake slams into the grid's edge or its own body. The player's goal is to eat as much as they can in order to get the highest possible score.

Use Cases:

UC1 - Game Start: The user starts the game, the Owl and the first food item are positioned randomly in the grid, and game systems and logic are set up.

UC2 - Move Owl: The user uses controls (typically arrow keys or swipe gestures) to change the direction of the Owl's movement and be able to move across the grid.

UC3 - Eat Food: If the Owls head reaches the grid cell containing the food, the length of the Owl increases and the score increments. A new food item is placed randomly on the grid.

UC4 - End Game: The game ends when the Owl hits the border of the grid or its own eggs or obstacles placed randomly within the area. The final score is displayed to the user.

UC5 - Pause/Resume Game: The user can pause the game at any time and resume the game from where they left off.

UC6 - High Score: The game keeps track of the highest score achieved on the device. If the user beats the high score, the new high score is saved.

UC7 - Game Speed: The user can adjust the speed of the Owl, making the game easier or more difficult.

UC8 - Restart Game: After the game ends, the user can choose to start a new game. This will use the same logic as the Game Start use case.

High-Level Functional Requirements

1. **Game Board and Owl Rendering:** The game should provide a visually appealing and responsive game board that fits within a normal 16:9 aspect ratio browser window. The owl's movements, as well as the appearance and positioning of the worms and obstacles, should be rendered accurately and clearly on the screen. Priority: 1

2. **User Input and Control:**
The game should handle user input for controlling the owl's movements. The player should be able to control the owl using arrow keys or WASD keys, and the game should respond to these inputs accurately and promptly. The owl's movements should be smooth and intuitive. Priority: 1

3. **Collision Detection and Game Logic:** The game should implement collision detection to determine if the owl collides with the game board boundaries, eggs, or any fruits or obstacles. When the owl collides with the boundaries or eggs, the game should end. When the owl collects a worm, it should grow in length, and the player's score should increase accordingly. The game logic should handle these scenarios and update the game state accordingly. Priority: 1

4. **Game Over and Restart:** When the game ends, either due to the owl colliding or the player choosing to quit, the game should display a game-over message and the final score. Additionally, the game should provide an option to restart the game, allowing the player to play again without having to reload the page or navigate through menus. Priority: 2

5. **Multiplayer Capability:** The game should provide multiplayer functionality, allowing multiple players to simultaneously connect and play the game together in real time. The multiplayer feature should support both local and online multiplayer modes, enabling players to compete or collaborate with each other. Priority: 3

Non-Functional Requirements:

1. Performance
    a. The game shall have responsive controls to ensure smooth and enjoyable gameplay.
    b. The game shall support a large number of concurrent players in the multiplayer mode without significant performance degradation.
    c. The game shall load quickly, minimizing the time it takes for players to start playing.
2. Reliability
    a. The game shall have minimal downtime for maintenance or updates to ensure uninterrupted gameplay.
    b. The game shall handle errors and exceptions gracefully, providing appropriate feedback to players.
    c. The multiplayer mode shall have mechanisms in place to prevent cheating, such as detecting and penalizing unfair play. (TBA)
3. Usability
    a. The game's user interface shall be intuitive and easy to navigate, catering to players of different skill levels.
    b. The game shall provide clear instructions and tutorials to help new players understand the gameplay mechanics.
    c. The game shall support multiple input methods, such as keyboard and mouse, and touch, to accommodate different devices.
4. Security
    a. The game shall protect user accounts and personal information from unauthorized access or data breaches.
    b. The multiplayer mode shall implement secure communication protocols to ensure the confidentiality and integrity of player interactions.

   c. The game shall have mechanisms to prevent and detect cheating, such as unauthorized modifications or hacks. (TBA)

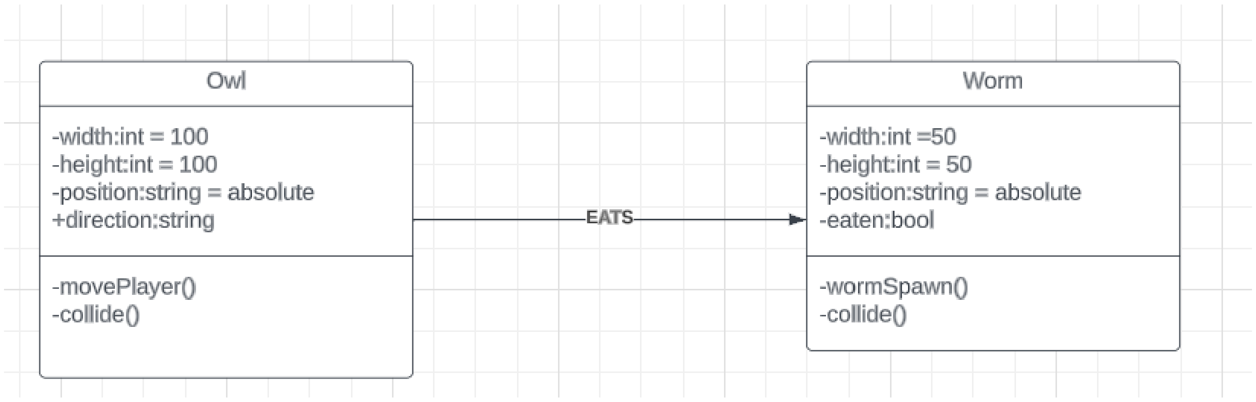
5. Maintainability
   a. The game's codebase shall follow modular and well-documented design principles to facilitate future enhancements and bug fixes.
   b. The game shall have a version control system (using github) in place to track and manage code changes.
   c. The game's development process shall include proper testing, debugging, and error-logging mechanisms to facilitate maintenance.
6. Scalability
   a. The game shall be designed to handle an increasing number of players as its popularity grows.
   b. The multiplayer mode shall support matchmaking and balancing algorithms to ensure fair and enjoyable gameplay for all participants.
7. Compatibility
   a. The game shall be compatible with major web browsers such as Google Chrome, Safari, Firefox, and Microsoft Edge, ensuring a consistent experience across different platforms.
   b. The game shall support various screen sizes and resolutions, adapting to different devices and display configurations.

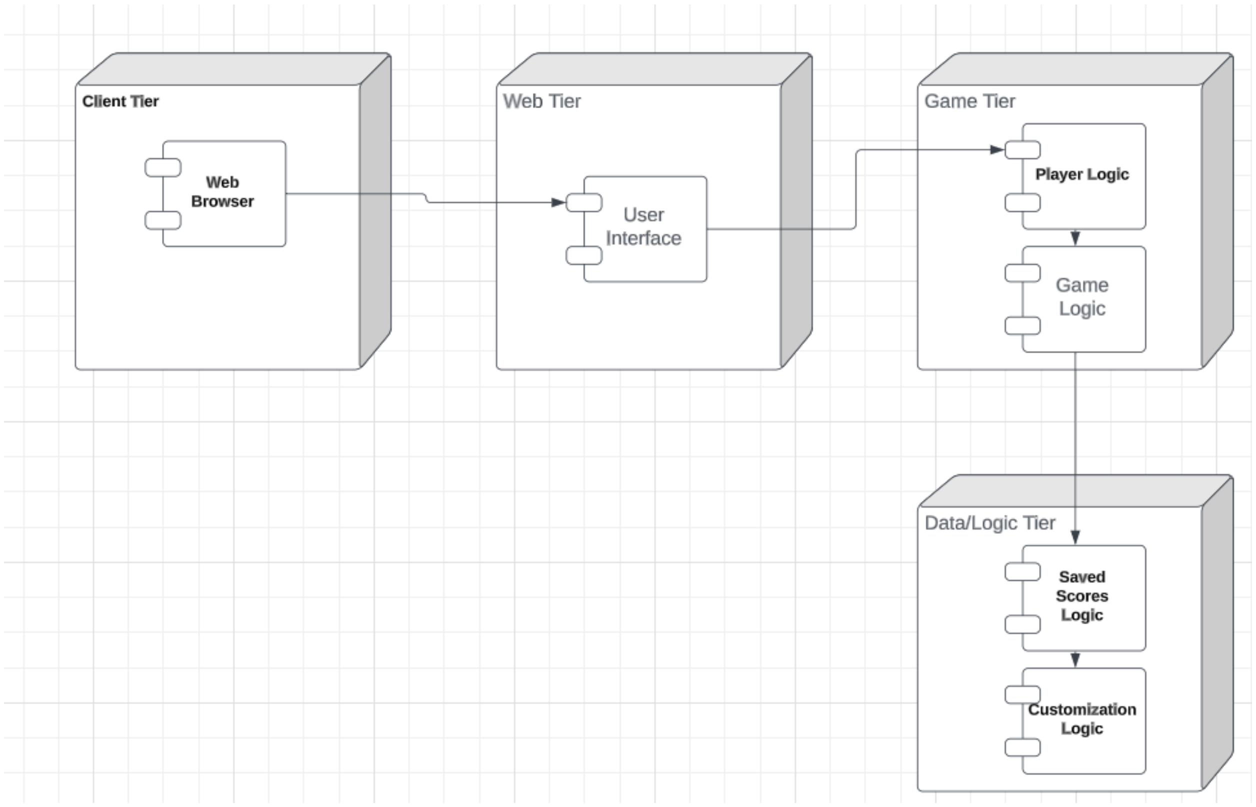
High-Level System Architecture and Database Organization:

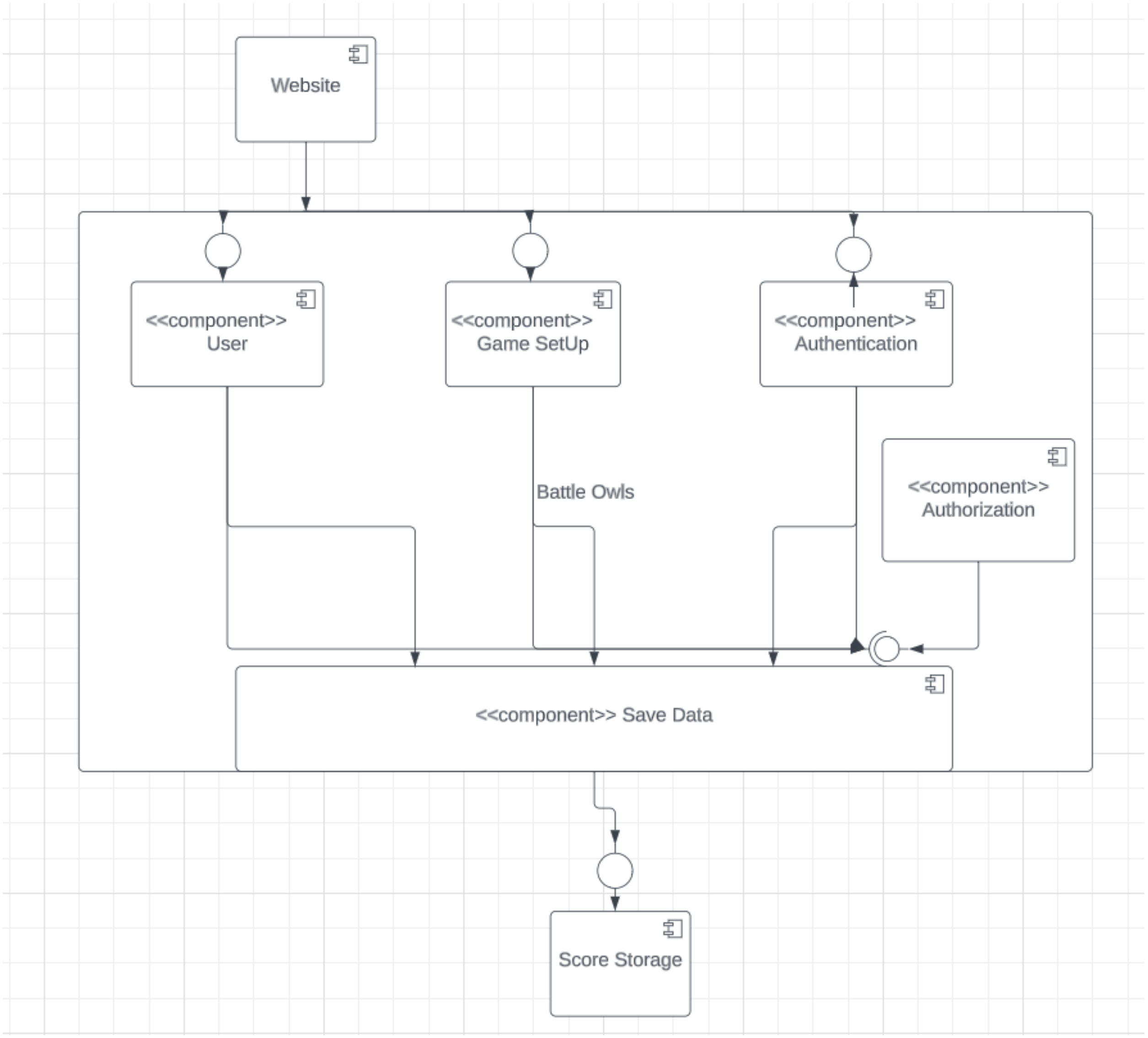
1. **Github:** Main source to transfer code and other files needed to have the application running. Also used for file comparison to prevent conflicts.
2. **Discord:** This is going to be our main application for communication for project development. All of our group's communication will be through this app.
3. **Browser Compatibility**: The system will be a web-based web app that operates on at least two of all of the major browsers, including Google Chrome, Mozilla Firefox, Safari, Opera, and Microsoft Edge. It will have functionality in it that will provide alternatives if the browser does not have JavaScript installed on it.
4. **Replit:** Where we will be hosting the browser game in order for people to play the game.
5. **HTML:** Coding for setting up website framework.
6. **JavaScript:** Coding for setting up the logic behind the game.
7. **CSS:** Coding for setting up classes and framework for the images to work and move properly within the website.

The database will be an organized leaderboard where the highest scores will be displayed at the top and it will go down in decreasing order. We are not using any APIs as this is a completely self-contained web browser game. Media will be stored within the files of the game itself not in any extended database. We have no way to search through data other than accessing the highest scores and manually searching through the order from highest to lowest. The ranking is based on the points accumulated during the game (eggs).

High-level UML Diagrams:




Client Tier
Web Browser
Web Tier
User Interface
Game Tier
Player Logic
Game Logic
Data/Logic Tier
Saved Scores Logic
Customization Logic


Website
<<component>> User
<<component>> Game SetUp
<<component>> Authentication
<<component>> Authorization
Battle Owls
<<component>> Save Data
Score Storage

Actual key risks:

1. Skills risks- We have the right skills to be able to make the framework and all of the coding behind the game to be able to make it work as we intended. However, we don’t necessarily have the full experience of making a project like this before.
2. Schedule risks- We do not have a large amount of time left for this project and even though under concept it shouldn’t take too long to create we need to pick up the pace or else some of the functionalities will be left out near the end.
3. Technical risks- We have not completed the project yet and if we don’t finish the project beforehand we won’t be able to test if our product works on different devices.
4. Teamwork risks- We have been procrastinating different parts of the project and we need to get the team together in order to finish different components that need to be done before we can progress later into the project.
5. Legal/content risks- We won’t have much of these because it is going to be using our own assets in order to design the game and the idea however used similarly before in Snake is original to avoid complications. The only potential issue is that the game works similarly to Snake.

We plan to resolve risks by starting on the final portions of the project earlier to give ourselves a bit more time to look things over. We will also be searching up some code for the game if we are struggling to complete certain portions of the logic. And finally, we will be using our own assets and putting our own twist on Snake so that it would not have any legal issues.

GitHub link:
https://github.com/Phoenixflare1579/Owl