# Requirements Analysis Document

## Broad-View Summary

The team is planning to build a Queen's edition of the classic game Battleship. Instead of dropping missiles to destroy the opponent's ships, this version of the game will require players to throw gentian dye at various vehicles parked in the Tindall Field parking lot using a mouse and keyboard. The target audience is Queen's University students (typically aged between 18 and 25 years old). The game itself will begin with a preparation phase, in which one player at a time will place a given number of vehicles on their board. These cars, buses, and bicycles each occupy a different number of spots on the game board. They will be using the mouse to drag and drop them, as well as the space bar to rotate their vehicles if desired. After confirming both users are ready, the action phase is initiated. The game will alternate between players, allowing them to drop a projectile at specific coordinates over their opponent's parking lot using the mouse. The chosen location on the board is marked as either a "hit" or a "miss" depending on whether a vehicle was present. The status of that spot now remains visible to both players for the rest of the game. Additionally, when an entire vehicle has been hit by gentian, users are notified with an animation and a special marking on those coordinates. When one player has managed to cover the entirety of their opponent's vehicles with dye, the game ends and that player is declared the winner.

## Adjustable Parameters

Players will be able to adjust the following parameters before beginning a game:

- Board Size*: The size of the map that the player and their opponent can place their ships (cars).*

Players will be able to adjust the following parameters after beginning a game, before the first turn:

- Ship (car) Placement: *Where on the map that the player's ships (cars) are individually placed, as well as their orientation.*
- Ship (car) type: *Which ship (car) they are placing before the first turn.*

## Functional Requirements

- As a player, I want to be able to play against another player on the same laptop so that I have an opponent.
- As a player, I want to be able to click a "Start game" button so I can begin playing Battleship against my opponent.
- As a player, I want to be able to quit the application whenever I want so I can do something else on the computer if I desire.
- As a player, I want to be able to place all my battleships (oriented horizontally or vertically) on the grid so I can proceed to play Battleship against my opponent.
- As a player, I want the location of my battleships to be hidden from my opponent, so my opponent is forced to guess where my battleships are located.
- As a player, I want to be able to select a square on the grid to shoot so I can attempt to "sink" one of my opponent's battleships.
- As a player, I want to know whether my last shot was a hit or a miss so I can know if I should shoot in that area again.
- As a player, I want to know whether my last shot sunk my opponent's battleship so I can know if I need to continue to attempt to sink their battleship or not.
- As a player, I want to know when my turn starts and ends so I can know when I must decide what square to shoot.
- As a player, I want to know if all my battleships have been sunk so I can know if I lost.
    - *Occurs before "Game Over" screen
- As a player, I want to know if I sunk all my opponent's battleships so I can know if I won.
    - *Occurs before "Game Over" screen
- As a player, I want the game to terminate when one of the players wins so I can know that the game is over.
- As a player, I want to be able to click a "Play again" button once the game is over so that I can start another game if I desire.
- As a player, I want to see who won the game once the game terminated so that I remember who won the game.

## Non-Functional Requirements

- As a player, I want to be able to enter my desired name so I can be identified while playing the game.
- As a player, I must always be able to access the rules within one click.
- As a player, I never want to be able to see my opponent's screen until the game is over
- As a player, I don't want my opponent to take very long turns.
- As a player, I want the game to have a fast loading time (less than 100ms).
- As a player, I want the game to be programmed in c++14 and compiled as a Windows executable.
- As a player, I want the game to use less than 2GB of RAM.
- As a player, I don't want the game to crash when I play.

## GUI Interface

Description of how the user will navigate through the Interface: The User Will start at, the game Menu (figure 1). From here he will have the options to start a new game, access the help/rules menu or access the options menu to configure game controls.

**Play Game:**

Upon clicking the sub-menu play game, the users will be asked to enter the usernames and then redirected to a screen (figure 2). Player 1 begins their turn first by placing their "battleships" on the 8X8 grid. The device is then passed to Player 2. Player 2 will be presented with a similar screen (figure 2) where they must also place their battleships. Next the players alternate on offence attacking the opponent's board. When one encounters a hit, a purple splash will be marked on the board. If the player misses it will be marked with a white splash. When a player wins a window will pop up congratulating the player on their victory (figure 5). In order to prevent cheating (seeing the opponents ship placement), after a player has finished their turn a screen will appear with the message “Pass the Device to player 2” (figure 4). Another alternative would be to not show the placement of the battleships at all after the initial setup. Additionally, a timer may be implemented so that a player does not take too long on the attack and to help accelerate the game.

## Help/Rules

The Help/Rules sub-menu will contain the following: an explanation on the rules of battleships, an explanation of the different “Battleship Modes” featured within the game, along with an interactive tutorial and or demonstration on how to play the game. The explanation of the rules and gameplay would be straightforward and presented with bullet points as follows:

Rules:

- The Object of the game is to sink all of the opponent’s pieces
- Each player has their own grid with their pieces shown, and a copy of their opponents’ grid with their pieces hidden
- At the start of each Game, Each player places their 5 pieces on their grid horizontally or vertically
- Alternating turns, players attack each other by selecting an area on the opponent’s grid that they would like to attack
- A purple splash of gentian violet will be displayed to mark a hit else a white marker will display a miss
- Once all the holes in an opponent’s piece have been filled the piece is considered “sunk”

Lastly, it would be beneficial to either have an interactive tutorial showing the player’s how the game is played or a simple GIF showing users the basics of how to play.

## Options

Under this sub-menu, users will be able to toggle the various setting for the game. Users will be able to adjust the number of Battleships, the size of the grid, timer, and include the option of power-ups. Along with the classic Battleship, users will be able to play an alternative with "Power up" squares. These squares will be hidden on the grid and if a user selects that space to attack, he will be given a power-up. Power-ups include but are not limited to: seeing the location of an opponent’s battleship, getting a second turn etc. There will also be options to toggle sound effects and background music.


Figure 1: Sample Game Menu for the Game

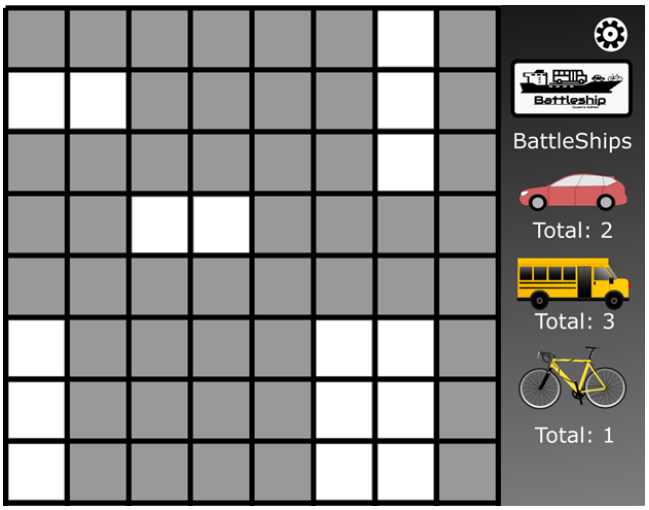


Figure 2: Sample screen of placing Vehicles

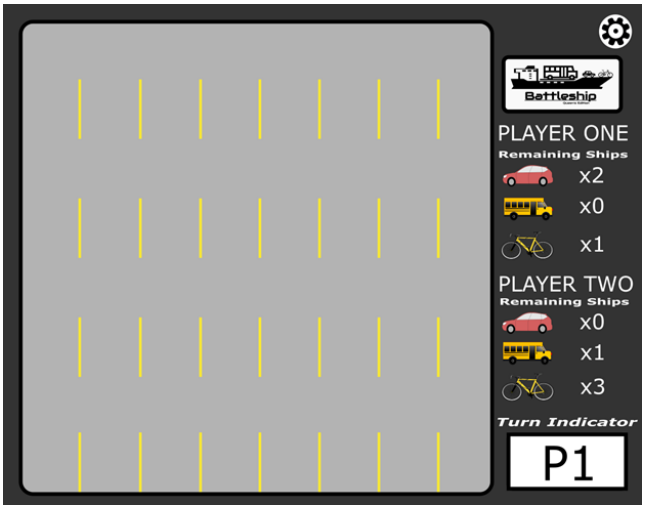


Figure 3: Sample Game Play screen of the user 1.

Figure 5: Winning screen displayed at the end of the game

## Team Roles:

| Role | Team Member Name |
|---|---|
| Team Lead | Isabelle Quail |
| Primary Software Architect | Danielle Mott |
| Illustrator | Jeremy Browne |
| Tester | Drew Anderson |
| Component Programmer | Douglas Gowing |
| GUI Lead | Eric Leuty |