SWE4203 - Project Milestone #2

Robert Greenan	3667592
Ben Jacobs	3663157
Thomas Donovan	3701046

1. Create Testing Plan
    a. #1/#5 - Win Condition for game failing:
        i. Given the current player has two symbols in a row, when they place a third connecting symbol, then their display should update to indicate win.
        ii. Given that no sequence of three is created, when the last tile is placed, display that there is no winner and handle end game conditions.
    b. #7 - Resetting does not close game
        i. Given a hosted game without an opponent, when the host copies the access code and clicks the reset then tries to join a game with the copied code, then they should not be able to join that game. It should close the game and tell the player the game is invalid
        ii. Given a started game with 2 players, when either player clicks the reset button, then the game should end and that be communicated to the remaining player.
    c. #3/#6 - Losing turn
        i. Given an error for a wrong move error is thrown by the program, when the user attempts to make a move where another player has already gone,, then the error should be displayed that the space is taken and the game should wait for valid input from the user.
        ii. Given an error for a wrong move error is thrown by the program, when the user attempts to make a move where they have already gone themselves, then the error should be displayed that they have already moved there and the game should wait for valid input from the user.
        iii. Given an error for a wrong move error is thrown by the program, when the user somehow inputs something unexpected, then an error should be displayed for invalid input and the game should wait for valid input from the user.

2. Identify a Candidate Impact Set
    a. Direct Impacts:
        i. Source Code
            1. CheckWon() within Game Class
            2. Play() within Game Class
            3. Move() within GameManager Class
            4. Each of the above will be directly impacted by changes made in the codebase when implementing this fix.
        ii. Game Behaviour
            1. A portion of the game behaviour should be affected as we believe that issue #5 is the main reasoning behind issue #1 being displayed incorrectly. This influence in the end should be positive by fixing another existing issue.
    b. Direct Impacts:
        i. Source Code
            1. New function resetGame() within GameManager Class
            2. main() within Main Class
            3. resetGame() within index JavaScript
        ii. Documentation
            1. New endpoint resetGame within API documentation
        iii. Game Behavior
            1. Clicking the reset button will end the current game that is being played, removing it from the servers list of active games
    c. Direct impacts:
        i. Source code
            1. PlayResult play() will be directly affected as the code in the function would be modified.
        ii. Game behavior
            1. Game behavior will change to no longer lose sync with the game-state when the player makes an improper move/input.

3. Perform the Necessary Code Modifications
    a. https://github.com/rgreenan55/SWE4203Group/pull/1
    b. https://github.com/rgreenan55/SWE4203Group/pull/3
    c. https://github.com/rgreenan55/SWE4203Group/pull/2

4. Compare Actual Impact Set with Candidate Impact Set
    a. Based on our analysis for Issue #1, the CIS appears to be identical to the AIS. The only unseen code modification that had to be made was the addition of a won playstate vs a game finished play state. Through various manual tests and thorough investigation, as of now, no unseen consequences have been made due to the changes made implementing this fix.
    b. Based on our analysis for Issue #7, the CIS appears to be identical to the AIS, all suspected entities were affected and no additional entities were. No unanticipated ripple effects have been discovered yet
    c. Based on our analysis for Issue #6, the CIS appears to be identical to the AIS. The only unseen code modification that had to be made was when the game swapped the turns of the players. Through various manual tests and thorough investigation, as of now, no unseen consequences have been made due to the changes made implementing this fix.

5. Reflect on how System Architecture Affected Process

Due to how the Javascript front end and the Java back end interact, updating information within a player's game that isn't the one currently acting is difficult and appears to cause issues beyond the issues we had listed in the previous milestone. There are likely ways to communicate bi-directional already, but better defining the functionalities and indicating their purpose more clearly through beacons and comments will allow future maintainers to easily comprehend what is going on and implement changes with more ease. This issue is consistent through all three issues that were solved.