Group Name: ConnectTechies
Members: Alisha Varma, Amishi Gupta, Ravnoor Bedi

Test Plan

To test the functionality, usability, and performance of our game, Connect 4 All, we wanted to test various edge cases and normal-case test cases for all of the functions we could to ensure high coverage and eliminate the chances of any errors when users play our game. This includes testing the game logic, user output/interface, performance under various circumstances, and error handling. With a game like Connect 4, there are various combinations of possible scenarios that result in wins and losses, and we tested a variety of these, such as winning in all possible directions with each player (Red and Blue). Additionally, we tested how errors are handled, such as when trying to enter a token in a column that doesn't exist and trying to enter a token in a column that is full. We also tested game statuses at different states of a game board, asserting that the status returned is correctly identified. We additionally tested player score tracking, ensuring the score increments correctly upon a player winning. Other than these main functionalities, we tested that the various helper functions that were used in our .ml files worked correctly, such as checking a board's score, copying a board, and choosing a valid random column. We omitted testing the implementation using the timer to return statistics of the average move time of the game round. This is because this would be tricky considering the time taken by a player to finish a game varies by milliseconds and would be challenging to test. As a result, we manually tested this by starting a timer on our phone when a game round began and ended it when the game round ended, and checked whether the average move time calculated by dividing by the moves was the same as the average move time that was displayed in the terminal. We also manually tested that the game board displays as we designed it, with a border around it. Although we were able to test that the game board's inner elements display correctly through our OUnit suite, we had to manually see that the border displayed as we wanted around the board's elements in the terminal (because the code displaying the border was not in the same function as the board's elements). In our test suite, we tested each of the functions in easy.ml, medium.ml, minimax.ml, and state.ml modules. This allowed us to test the full functionality of our game by covering all of the modules (other than main.ml) which were used to create it. Our testing approach demonstrates correctness of the system by making sure we tested various different combinations to ensure high coverage of most of the functions used in the program. This allowed us to verify that everything worked as intended.