# Final Project Report

## Overview

Our final project was the creation of Shadle, a color-based guessing game implemented in Python using the Pygame library. The game challenges players to guess a randomly generated color sequence within a limited number of attempts.

## Team Contributions

Allison: I focused mostly on the generate_sequence() and check_guess() functions. They worked in tandem to create the sequence of colors to be guessed and then compare that to the user guess. Additionally, they were called by the enter button to display the accuracy of the guess. I also had a hand in the button loop, specifically updating user_guess and the empty_space child classes when the color, enter, and delete buttons were pressed.

Addy: I focused mostly on the display, the game_intro() method, the lose method, and some debugging. The game intro begins with a start button which goes to the main loop, and a quit button, which quits the game. The lose method counts the attempts the user makes and if it is over 6, the user loses. I also debugged the button_loop() so that the number of squares the user clicks cannot be greater than 4.

Ray: I did both the Button and Empty Space classes. Both classes serve as visual representations for different interactable buttons in the game along with just representing place holders. These classes do not hold much code, they do lead to a large quantity of code being required for the game to run and look polished. So, I focused a decent amount of time on the visuals of the game. I made the keyboard buttons correspond with the buttons' actions in the game. This means that pressing delete on the keyboard will delete the colors selected in the empty spaces.

## Original Plan vs. Final Product

Our original plan was to have six colors and a four-color sequence to be guessed. We planned a function to generate the sequence, buttons for each color as well as enter and delete, and a function that checks the user guess and displays borders in colors that indicate accuracy. Additionally, we planned a welcome screen, a win screen, a loss screen, a score, instructions, and a game loop. The classes we planned were Button and Indicator.

We did have six color options and a four-color sequence. We also had a function that generated the sequence and checked the user guess. One change was the guess accuracy was displayed using text as opposed to colored borders, just because it was more reminiscent of the Mastermind game. Thus, we never created an Indicator class. We still created our planned Button class, and made child classes of colors, enter, delete, play, and quit. Also, we created a welcome screen, win screen, loss screen, and a game loop to facilitate the entire game. Another thing we did not include was a score, because it seemed non-essential to the game and the original wordle does not have scores. Also, we made the delete and enter buttons accessible on the keyboard as well, which was not in our original plan. Finally, we put the instructions in the README file instead of within the game because it seemed redundant to do both.

## Reflection on ChatGPT

We used ChatGPT to assist in debugging our code. Specifically, there was a loop that was only going through the first index of a list and then stopping. In using ChatGPT, we discovered that there were too many loops nested within one another, so some things needed to be separated. It was helpful in the end, but it took a lot of prompting to come to the correct solution.

## Challenges

One of the challenges we faced was working within the terminal and a joint repository. There were some difficulties in pulling, committing, and pushing files, especially when we encountered conflicts. It was also slightly confusing to use branches at first. A workaround that we used twice was simply copying and pasting code we wanted to push into a new file when a file either would not commit or push. Also, working in the terminal and within branches is a skill that required time to become proficient at. In the beginning it was a hassle, but by the end of the project it was automatic and easy.

Another challenge was the implementation of pygame. There were initial issues with downloading the library and getting the display to show up when running the code. Then, all the possibilities were overwhelming, but with time it became much easier. Like terminal, it had a learning curve and time and practice was the solution.

A final challenge was working on the project at the same time. We would often assign aspects of the project to do before our next meeting, but sometimes one could not work without the other. Then, the person with the later facet of the project would have to wait on the earlier part and could not do any work in the meantime. The solution to this was just being clearer about who was doing what and when, as well as assigning non-overlapping parts of the project.