Yasmin Senior

Fall 2022

Software Engineering

Assignment #3

Code Review

When given a Boggle board and a dictionary, every particular solver's aim in the game Boggle is to uncover and compose as many words as possible. This dictionary offers an alphabetized list of valid words that the game's solver or solvers should look for when studying the Boggle board. Players, or solvers, can approach this challenge in a variety of ways. My partner Chad Toomer decided to create his own Boggle solution using a hash map.

Chad's code is flawless. It's not just exceedingly tidy, but it's also legible, with descriptive variables and function names that clarify how to utilize them. He made it a point to check the board for consistency before attempting to solve a Boggle board. Incongruous grids have the letter 'S' followed by the letter 'T' or the letter 'Q' followed by the letter 'U'. Chads design contains built-in checks and balances for the authenticity of its boards. When lonely 'Qs and 'Ts are recognized, his approach snuffs them out and returns false. Given that sound grids are N x N, his method also assures that the input from the boards follows that model. If a grid is found to be faulty, his loop ensures that each row and column are the same length and returns an empty Solutions list. Chad may add a print statement telling users that their boards are invalid to this portion of the code to improve it. The current setting of the returned solutions list might be unsafe.

After the grid is certified, the developer should rewrite the dictionary and grid into lowercase for future programming simplicity. To avoid future code roadblocks, they should ensure that all strings in the board and dictionary are in the same case. His code correctly transforms the grid and dictionary into a single case. His keen eye is on display here. It is crucial that they foresaw the problems that may arise if the board and dictionary were not in the same case.

To find the solutions in the Boggle board, Chad constructed a collection of solutions and a hash map for the dictionary. His software loops around the N x N grid, populating the solution set using their function to find words. His arranges her code neatly by making optimal use of the existing comments. This is crucial because other engineers, even if they are inexperienced with his code, will be able to understand it. The use of descriptive functions and variable names also helps other developers comprehend their code. In terms of formatting, I don't believe there is much code to enhance here. They do an excellent job with this throughout all lines of code.

Chad utilizes several minor aid tasks. The auxiliary functions used to supplement find Words add to code clarity as well. His knowledge also helps me as a developer since if a helper function fails, he understands exactly where in the code to focus on debugging. This benefits the developer since it allows them to spend more time debugging rather than looking for bugs.

Finally, Chad excels in writing clean, simple, and understandable code. This is made feasible by his usage of pre-existing comments, descriptive variables, and function names. Chad might enhance his code by printing an error notice when the grid becomes incorrect. His testing should also be improved. While the testing document contains solid coding principles from boggle-solver.js, He might create tests to demonstrate that the solver works for grids of various sizes. Otherwise, the check for incorrect grids suffices. As well as constructed and tested the boggler solver successfully.

Assignment 3

https://github.com/chadtoomer/Software_Engin3