Isabel Hughes
CS 555
11/6/2023
Assignment 2 - Refactoring

1. List a minimum of 5 distinct code smells you've learned from the class and provide a brief explanation and/or your experience for each.
    a. Bad Method/Variable Naming
        - Something I frequently do while coding is poor variable naming. This can lead to not knowing what a variable is later on in code development, and can cause some major issues if someone else tries to work on your code. I very commonly name variables ‘x’, or something else unidentifiable. I also do this with helper functions by just calling them ‘helper’. One of the things I can fix in my work from Sprint 2 is clarifying variable and function names.
    b. Excessive Comments
        - Something else I frequently do is write excessive comments. This is because I nest code, or just over-explain simple lines like x+=1. This is bad for our code, as it aims to explain lengthy and confusing code. Code that is nested should be turned into separate expressions. Similarly, I make big blocks of comments to explain certain parts of code, which should be moved to be more explainable methods.
    c. Too large classes/methods
        - Large classes/methods can often be confusing, as trying to edit any part of them is very likely to give your code bugs. I can say that our PostUploader.js file is very lengthy, and could be split up into smaller pieces to make it less difficult to edit. Hopefully our refactoring can help to make that feature less “smelly”.
    d. Complexity (code takes longer to run due to complexity)
        - Searching through lists or through our “database” currently takes a high complexity factor, which may make the loading of family members or logging in take longer. This isn’t great for end users or for developers testing the code. Optimization of these algorithms for finding family members and logging in will hopefully streamline the code testing process.
    e. Dead Code (unused methods/classes)
        - Dead code can make your methods and classes jumbled with unused functionality, and be confusing later down the road for other development team members. Removing/commenting out unused code will allow other members to work without worrying about that pieces functionality, and also prevent the website from being vulnerable later on. (Return2Win attack on unused functions to run malicious code).

2. Identify two different bad smells in one or more of your user stories. If your code is already pristine and has no bad smells, then add bad smells that can be refactored.
    a. In the Login.js file, we have a piece of dead code that was originally used, and now it is unused. This also comes along with a variable which is being changed in cases where it does not need to be changed. To fix this, I am going to fix where the variable is changed and remove the piece of code that is no longer used. This does unfortunately remove one test case, as we need to navigate to another page to check for a successful login, and I believe that cannot be tested.
    b. Also in the Login.js file, I have some functions that I feel could be better named for navigation. handleClick doesn't tell anyone what a function does, nor are there any comments to explain it. To fix this, I will be renaming the functions and adding in comments to explain anything that may be confusing.