2a. The app, *Stat Tracker*, was developed by me and a partner. The app was made using the MIT App Inventor, a block-based programming web application. The purpose of this app is to record your own baseball statistics. With *Stat Tracker* you can record statistics in batting, fielding, and pitching. In batting, you can keep track of your hits, strikeouts, walks, ball-in-play outs, singles, doubles, triples, and home runs. The app calculates your batting average and slugging average with this data. The fielding screen allows you to input your errors, putouts, and assists. This also keeps track of your total chances, and thus calculates your fielding percentage. Lastly, the pitching screen allows you to keep your pitch count; add strikes, balls, strikeouts, walks, let up hits, earned runs, innings pitched, and batters faced. The app then can calculate your ERA and your strikeout to walk ratio (K/BB). Every screen also has a reset button that resets all statistics on the screen you are on.

2b. In our app, there are multiple equations that divide variables by other variables. Because of this, sometimes the divisor variable can be equal to zero, causing the app to try to divide by zero. This resulted in an error message being displayed until the divisor is greater than zero. I came up with the algorithm that prevents the app from trying to divide by zero. Division will not occur until the divisor variable is greater than zero. This algorithm can be found in many different places throughout the app, and is even in a few procedures as well. At one point in development, we had issues coding the slugging average formula. The formula for slugging average is (1B+2B*2+3B*3+HR*4)/Number of At-Bats. We struggled finding an easy way to set up that formula in App Inventor. However, we solved this issue by creating a slugging average variable, and by figuring out that we could multiply and add together everything in just one line. This made our app very concise and less complex.

2c. On the Defensive Statistics screen, fielding percentage is calculated. The fielding percentage formula is (Total Chances – Total Errors)/Total Chances. On the screen, there is a list picker, labeled “Fielding Result”. When you open up the list picker you can choose from three options: putouts, assists, and errors. Choosing any one of those results in that variable being increased by one, and setting its respective label to its new value. However, choosing any of them will always result in the variable “chances” to be increased by one. The program then sets the variable “fielding percentage” to (chances – errors)/chances. The algorithms that increase errors, putouts, assists, and chances by one work well alone and display statistics that most people are familiar with. The incorporation of fielding percentage allows for users to have something to strive for, getting their fielding percentage as close to 100% as possible, which makes its incorporation very important in the app. As much as this app is intended to record statistics, it should also serve as a source of motivation.

2d. On the pitching screen, two different values are calculated, ERA (Earned Run Average) and K/BB (Strikeout to Walk ratio). ERA is the amount of runs the opposing team scored while you were pitching times the total amount of innings in your game (9) divided by the amount of innings you pitched. This is calculated in the app whenever you add one to either innings pitched or earned runs. This results in the same thing happening in two different places, which is quite redundant. To fix this, we made a procedure. This procedure incorporates something mentioned earlier, the algorithm that eliminates dividing by zero. It also performs the ERA formula, sets the

ERA label to the answer of that formula, and stores that value in a TinyDB. K/BB is the amount of people you struck out divided by the amount of people you walked. This formula occurs in two places as well, when a strikeout is added or when a walk (base-on-balls) is added. To remove redundancies we turned the whole process into a procedure. This procedure also checks to make sure the divisor is not equal to zero. It then does the K/BB formula, sets the label to the quotient, and stores the value in a TinyDB. These procedures removed complexity from the app by removing redundant code from the main algorithms. It also helps if we ever find bugs since changing the procedure will fix bugs in both algorithms the procedure is in.