Test Report

Similar to our last iteration, our team did not have enough use cases for every team member to complete one from start to finish. We actually only had one use case, and also needed to implement the database. Zach was tasked with implementing and testing Use Case 5: Modify Base Data. Ethan was tasked with helping Zach with Use Case 5 documentation, and also saving and loading user information using the new database. Rachel was tasked with writing up the team test report and loading and saving reports from the database. Dan was tasked with implementing the database and database manager. Our team agreed on Thursday, 12/8/2016, as our milestone date when all of our code would be committed for other team members to test.

All unit tests that we made for all of our main algorithms were able to pass. Use Case 5: Modify Base Data was located in the config controller class. This class already had unit tests associated with it. The equivalence partitions for testing this use case would be making sure any input that the user inputs into the config gui would be able to be saved into the database. The gui itself verifies for incorrect input, such as statistics that are too low for the simulation. We would also need to verify that the preset is saving all valuable information for that specific user. Different users should be able to have different presets saved for them. Boundary values that need testing would be values such as zero or negative decimal amounts, but once again, the gui checks the values. There are no string input values, so there is no need to test for those inputs. All values being saved to the database for presets are integers. The cyclomatic complexity for this use case is three, and all of these decisions lie within the database manager class.

The integration of the different parts of our program worked well. This iteration, we didn’t have many parts to integrate. It was mainly integrating the use case discussed in the paragraph above and integrating the database. The tests we used for the database was simply

making sure the correct amounts were kept, and that the database only kept one record. We had six controllers including the database manager, which translates to six modules we had to integrate. However, we integrated these modules in previous iterations, and already tested for correct input and output. Using a module tree, the integration complexity of our program is seven. We would need seven integration tests to fully integrate our system. These tests included authenticating a user, passing correct user information to the config controller, loading and storing combat presets correctly to the database, passing correct preset information to the simulator controller, passing correct combat statistics to the combat round controller, passing the end of simulation statistics to the report controller, and loading and storing reports using the database. In order to run all of these integration tests, we followed data through the system to make sure it was either kept the same or changed when appropriate during combat, and then saved to the database if necessary.