That One Team

Test and Validation Plans

Adam Kowalchyk
Daniel Hoberman
Maya Fleming
Trevor Diuco

Course: CPSC 224 - Software Development

Instructor: Aaron S. Crandall

I. Introduction

I.1. Project Overview

The software to be tested simulates a traditional game of Yahtzee. The software supports 1-6 players and includes an interactive Graphical User Interface. The program needs to be able to begin correctly and prompt the user with how many players they would like to play with. It needs to display 5, 5-sided dice using a random method and correctly calculate potential scores for each line of the scorecard. Finally, the program needs to properly store the sum score of each player and determine a winner.

I.2. Scope

This document will provide an outline of how testing will be conducted in order to determine fulfillment of our functional requirements. To test our program, both unit and functional testing will be executed. Unit testing allows us to validate the methods and attributes of each class and the functional testing will test the overall success of the program. In addition, the functional testing will help us check for and correct any issues with our user interface. Our goal is to create classTester files for each class which could be run to validate output of predetermined controlled input.

II. Testing Strategy

1. Identify changes implemented for player class
2. Identify which tests will be used to check implementation
3. Design test cases, look at data results for test cases
4. Identify expected results for testing
5. Perform tests
6. Analyze testing results
7. If tests are successful code is operating smoothly, run final performance tests
8. If testing fails, document specific data for testing failure. Upload for the group to analyze. Resolve problems when applicable.
9. Repeat throughout project development.

III. Test Plans

1. Develop new classes.
2. Create unit tests for each class
    a. Assign members of team to create unit tests for each class
    b. Other members should review the tests and apply fixes if needed
    c. Make sure that each class is working as expected before moving to new implementations
    d. If tests fail, analyze failure data, and fix code.
3. Once all coding implementations are complete and unit tests have been passed begin system testing
    a. Discuss with group expected testing data, and results.
    b. If tests pass move on to next stage of testing
    c. If tests fail, analyze failure data, and fix code.
4. Once system testing is complete, test if all functional requirements have been met.

    a. Discuss with group expected testing data, and results.
    b. If tests pass move on to next stage of testing
    c. If tests fail, analyze failure data, and fix code.
5. Once all tests listed above have passed, begin user acceptance testing
    a. Discuss ways to stress the system, to make sure code is completely safe.
    b. Have group test code to see if it fits users standards. Find friends not related to the project to test code.
        i. Discuss with group expected testing data, and results.
        ii. If tests pass move on to next stage of testing
        iii. If tests fail, analyze failure data, and fix code.
6. If all tests listed above have passed, assignment is likely complete and reaches course standards.

III.1. Unit Testing (Planned)

For unit testing, tester classes for each class will be divided between the group. This way, the other members of the group can look over the testing material to determine the quality of the testing as well as to verify the tested inputs yield the desired outputs. A final review of the testing files will be done, prior to submission.

III.2. System Testing (Planned)

For system testing, the program will be tested in order to evaluate the system's compliance along with the requirements needed for the program. This will be the one of the final tests performed in order to verify that the system is meeting the purpose of what the program is supposed to run and identify any faults in respect to the overall requirements.

III.2.1. Functional testing: (Planned)

Functional testing will require us as a group to interact with the GUI. By playing the game multiple times, we can test to see that all functionality of the program works as expected.

III.2.2. User Acceptance Testing: (Planned)

Acceptance testing and installation testing check the system against the project agreement. The purpose is to confirm that the system is ready for operational use. During acceptance tests, end-users (customers) of the system compare the system to its initial requirements (if necessary) with help from the developers. You could have a roommate, friend, or family member play your Yahtzee game and see if they are successful or expose any bugs.

Once all previous testing harnesses have been complete, begin open testing on Users of the program. Each team member should ask family members and friends to try out our yahtzee game. If small errors or mistakes are identified by the users playing the game, notify other team members to decide how to better implement the part of the program identified.