MSSE SOFTWARE, INC.

**Test Plan for**

**(GolfScore Release 1.1)**

**May 24, 2021**

Confidential and Proprietary Information of Datacard Worldwide

**By Egemen Bozkus**

Contents

1.0 Introduction 3

1.1. Objective 3

1.2. Project Description 3

1.3. Process Tailoring 3

1.4. Referenced Documents 3

2.0 Assumptions/Dependencies 3

3.0 Test Requirements 3

4.0 Test Tools 4

5.0 Resource Requirements 4

6.0 Test Schedule 4

7.0 Risks/Mitigation 4

8.0 Metrics 4

Appendix A – Detailed Resource Requirements 5

Appendix B – Detailed Test Schedule 6

# Introduction

## Objective

The Test Plan is an aggregation of information, which describes the entire test activity for GolfScore Release 1.1. It covers the entire testing effort (unit, development test, system verification test, and Beta). It identifies the product requirements, schedules, resource requirements (people, effort and equipment), quality, assumptions, exclusions, and risks.

A preliminary Test Plan is prepared for the Project Team during the System Phase of PEAQ Process. This Test Plan will be updated in the earliest possible time of the Implementation Phase, so that progress can be tracked during implementation.

## Project Description

This project is to build a software program that will process scores from golf tournaments to produce reports on the winners of the tournaments as well as performance data on the players on each course played.

This project will be written in C or C++ and should run on a PC with Windows 2000 or any later version. The program does not use a GUI but rather a command line interface.

## Process Tailoring

The only document referenced by this entire test plan is the Software Requirements Specification (SRS) for GolfScore Revision 1.1.

Specific qualities planned to test for in this test plan are specification, functional, stress, volume, destructive, and performance. Other qualities not listed will be justified below.

Compatibility is not worth testing for this project because this program is not intended to interact with any other system. It simply a program that reads and writes files which is designed to run on Windows only.

Testing documentation should be maintained but can be kept simple since this is a very basic and straightforward project.

# Assumptions/Dependencies

**Tournament Assumptions**

This subsection will list all assumptions regarding the golf tournaments. These are the game rules that this program must strictly follow. A golf player can only play a course once. The number of players in the tournament can range from 2 to 12. There are only 5 golf courses, and each golf course has 18 holes. A “par” for each hole can either be 3, 4, or 5 *strokes* (see further details on scoring in SRS section 2.3.2). A player’s *score* is calculated using their number of *strokes* over or under par to complete a hole. A player’s *stroke count* for a golf course is the total number of strokes made from 18 holes in that course. A player’s *score* for a course is the total scores from all the 18 holes in that course. A player’s tournament score is the sum of the scores from each course.

**Testing Assumptions**

All assumptions for the testing process are listed here. First of all, no behavior or error handling of the program, listed in the SRS (sections 2.3 – 2.6), should be assumed true and should be rigorously tested for any deviations from the stated behaviors. As with a standard testing process, there will be unit, design verification, system validation, and customer acceptance testing. Developers will have complete responsibility over unit testing and partial over design verification testing. Testers will have partial responsibility over design verification, complete responsibility over system validation and customer acceptance testing. Testers will be working directly with customers during the customer acceptance tests. There will be no regression testing as this is the first release of the program and prior tests do not exist. Currently there is no beta testing but this may become subject to change in the future.

**Testing Dependencies**

The testing process order is: unit > design verification > system validation > customer acceptance

**Program Usage Assumptions**

The program will be interacted with via command line instructions. The calling procedure goes as follows:

>**golf** *options filename output-directory*

Example instruction:

>**golf** *-ctg C:\golfdata\golf1.txt C:\golfdata\results*

# Test Requirements

|  |  |
| --- | --- |
| **#ID** | **Requirement** |
| #1 | Program does not accept any other file format other than .txt for input and displays an error message. |
| #2 | Program displays an error message if no input filename is given or if file does not exist. |
| #3 | Program displays an error message if the input text file does not follow input data conventions or game rules. |
| #4 | Program does not alter the input file in any way. |
| #5 | Program calculates score correctly with respect to the game rules and input data. |
| #6 | Program only generates the requested files stated in the command line. |
| #7 | Program displays complete help information upon request with no spelling or grammar mistakes. |
| #8 | Program displays an error message if *options* parameters are incorrect. |
| #9 | Program accepts multiple tags as an *options* parameter in any order/combination as long as they are c, t, and/or g. |
| #10 | Program generates the correct file matching the correct *options* tags. |
| **#ID** | **Requirement** |
| #11 | Program is able to locate existing output directories if given. |
| #12 | Program will output to the directory of input file if a directory is not given for *output-directory* parameter. |
| #13 | Program will overwrite output files if they already exist in the current output directory. |
| #14 | Program will not delete or overwrite any unrelated files or directories. |
| #15 | Program will ignore any parameters after *output-directory* and will not show an error message. |
| #16 | Program will use its own file path if no path was given for input file parameter. |
| #17 | Program does not crash computer. |
| #18 | Program does not take longer than 5 seconds to generate output files. |
| #19 | Program displays an error if the given data .txt file is incomplete or empty |

# Test Tools

|  |  |  |
| --- | --- | --- |
| **Tool Name** | **Information** | **Needs to be developed?** |
| CppUTest | Free tool based on JUnit for C and C++. | No |
| Golf Data Generator | Develop a tool that can generate valid input test data files in .txt format. This can be a single .cpp file that is programmed to write sample golf tournament data to a new text file in the correct format. | Yes |

# Resource Requirements

The only resources this test plan requires other than a testing tool is sample golf data to test with. There was no specific source of golf tournament data mentioned (such as a third-party database) so it is up to developers/testers to generate the necessary input data files for testing.

# Test Schedule

There was no overall project development schedule provided to refer to. However, this test schedule should be easily adaptable to any software development life cycle. Static analysis should begin at the early stages of the software development. Dynamic testing, starting with unit tests, can begin after static tests are finished. Each dynamic testing phase (unit, integration, system, acceptance) should take a week to complete, acceptance tests may take longer if needed due to scheduling with clients/users.

As mentioned under “Test Dependencies” in section 2.0, the testing process will follow a waterfall structure. The testing phase should begin once the tools in section 4.0 are acquired/developed. A week should be enough time to complete rigorous testing in each phase as this is a simple project. For reference, a waterfall method is where the next test phase waits to begin until the previous is finished. Customer acceptance phase may take longer than a week depending on customer’s schedule availability.

# Risks/Mitigation

This project is low risk by nature. The program is simple and is easy to test. The program is not meant to be used for any task that is critical (sensitive information, lives, etc.). The worst damage this program could possibly create is by crashing the computer or overwriting the wrong files. This can be mitigated if not entirely prevented through static analysis and dynamic testing. This program is also meant to run with no connection to internet which resolves many if not all security concerns.

# Metrics

The following metrics data will be collected. Some will be collected prior to, and some after product shipment.

**Prior to shipment:**

Effort expended during DVT, SVT and Regression

# of defects uncovered during DVT, SVT and Regression, and development phase each defect is attributable to

* Test tracking S-Curve
* PTR S-Curve
* Number of crashes

**After shipment:**

# of defects uncovered and development phase each defect is attributable to

Size of software

Appendix A – Detailed Resource Requirements

Test cases for this project are shown below. More test cases may be added during the testing process.

|  |  |  |
| --- | --- | --- |
| **Test Case #** | **Test Case Description** | **Test type** |
| 1 | A non .txt file input shall display an error | Functional |
| 2 | An empty input .txt file shall display an error | Functional |
| 3 | An incomplete input .txt file shall display an error | Functional |
| 4 | A command of ">golf -tcg (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 5 | A command of ">golf -gct (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 6 | A command of ">golf -tgc (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 7 | A command of ">golf -gct (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 8 | A command of ">golf -cgt (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 9 | A command of ">golf -ctg (valid txt filename) (valid directory)" shall generate 3 output files | Functional |
| 10 | A command of ">golf -tc (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 11 | A command of ">golf -gc (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 12 | A command of ">golf -tg (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 13 | A command of ">golf -gt (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 14 | A command of ">golf -cg (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 15 | A command of ">golf -ct (valid txt filename) (valid directory)" shall generate 2 output files | Functional |
| 16 | A command of ">golf -t (valid txt filename) (valid directory)" shall generate 1 output file | Functional |
| 17 | A command of ">golf -c (valid txt filename) (valid directory)" shall generate 1 output file | Functional |
| 18 | A command of ">golf -g (valid txt filename) (valid directory)" shall generate 1 output file | Functional |
| 19 | A command with missing directory shall generate output files in same directory of GolfScore program | Functional |
| 20 | A command with an incorrect output directory shall throw an error | Functional |
| 21 | A command of ">golf -h" shall generate help instructions | Functional |
| 22 | A command ">golf -hc" shall throw an error | Functional |
| 23 | ">golf -hfg" shall throw an error | Functional |
| 24 | ">golf -hctg" shall throw an error | Functional |
| 25 | ">golf -h (valid txt filename) (valid directory)" shall throw an error | Functional |
| **Test Case #** | **Test Case Description** | **Test type** |
| 26 | Spamming commands should not crash the program | Destructive |
| 27 | Outputing to a directory with same output filenames but different filetypes do no get overwritten | Functional |
| 28 | Spamming the same command Shall only overwrite existing output files | Destructive |
| 29 | A valid 3 file generate command will execute and complete in no later than 5 seconds | Performance |
| 30 | A valid command for 3 output files will display correct information about the golf tournament | Specification |