MSSE SOFTWARE, INC.

Test Plan for Golfscore

Version 1.0

31 August 2017

Contents

1.0 Introduction

1.1. Objective

This is the test plan for GolfScore Release 1.1. It provides information about the scope of test, the test methodology and the test timeline. The document also covers the test cases, test dependencies, required resources, test tools and the test metrics. The primary objective of this test is to verify the Software Requirements Specification in the GolfScore's SRS. This test plan is a living document that will constantly be updated to reflect according GolfScore releases.

1.2. Project Description

GolfScore is a program used to generate golfers' result of a golf tournament. This program takes an specified input text file and produces three output text files in a format suitable for printing. The program runs as a stand-alone executable. The program has no GUI and is executed via a command line interface, from within an IDE (Integrated Development Environment), etc. The program shall also handle errors in the inputs or outputs according to defined specifications.

1.3. Process Tailoring

The program will be developed in C or C++ that will be run on windows 2000 of later versions of it. Given the program runs in a standalone environment, it requires network or other systems. All the tests shall be conducted via the command line interface. The follow tests are covered in this test plan:

Verification Test: To ensure the requirements on the key functions of GolfScore based on SRS Version 1.1 is tested. The tests will be grouped into functional and non-functional testing. Functional test will include boundary value analysis & equivalence partitioning techniques to select cases that covers all the scenarios. Verification of the functional tests is done by checking the expected outputs of the program given the various different combinations of inputs.

Regression Test: To verify the functionalities of the program that are defined in the versions of SRS *before* Version 1.1. This is to make sure the new features to the program do not break the features that were built and tested earlier.

The following documents are used to create this test plan:

- Software Requirement Specification (SRS). Revision 1.1 18 July 2017

2.0 Assumptions/Dependencies

The assumptions made within this test plan includes:

- The functionalities of the modules were already tested by the development team with their unit test
- There are no Artificial Intelligence/Machine Learning functionalities required in this release version.
- The program release date is fixed on 16 October, 2017.

3.0 Test Requirements

See Annex C for the Test Cases.

S/N	Test Requirements	Reference to SRS
1	Test that the program can execute on a PC with WINDOWS 2000, XP, VISTA, 7, 8, 10 and 11	1.3
2	Test for all valid and invalid command line execution to the program	2.2
3	Test for valid and invalid Course Records with Delimiter Record	2.4.1 – 2.4.2
4	Test for valid and invalid Golfer Records with Delimiter Record	2.4.3 – 2.4.4
5	Test for valid and invalid number range of golf courses for a tournament	2.3.1
6	Test for valid and invalid number range for golfers entered in a tournament	2.3.1
7	Test for valid and invalid records for each golfer’s performance across the 18 holes between 3 – 5 strokes/par	2.3.1
8	Test for correct calculation of the golfer’s **score for each golf course** based on valid golfer’s records	2.3.1 – 2.3.2
9	Test for correct calculation of the golfer’s **stroke count** for each golf course based on valid golfer’s records	2.3.1
10	Test for correct calculation of the golfer’s **score for the tournament** based on each completed valid golf course	2.3.1 – 2.3.2
11	Test for correct data output (3 text files) from the program saved according to the expected location	2.5
12	Verify the output for the **Tournament Ranking Report** has a file name trank.rep, AND the content and formatting are as expected	2.5
13	Verify the output for the **Golfer Report** has a file name golfer.rep, AND the content and formatting are as expected	2.5.2
14	Verify the output for the **Course Report** has a file name course.rep, AND the content and formatting are as expected	2.5.3
15	For all invalid input parameters, verify that errors are reported with an appropriate message	2.6.1
16	For all invalid data inputs, verify that errors are reported with an appropriate message	2.6.2
17	If any of the requested output reports already existed, verify that the program is able to handle this appropriately	2.6.3
18	Verify that the program is able to finish the execution of ONE work load under 1 minute	4

S/N	Test Requirements	Reference to SRS
19	Verify that program is able to pass all Regression test cases	NA

4.0 Test Tools

The following test tools will be used to support the test activities:

Tool Name	Purpose
Atlassian JIRA	Bug Tracking
qTest	Test Management and Design
JUnit	Functional Test, Regression Test
JMeter	Performance Test

5.0 Resource Requirements

The following resources are needed to support the tests:

- 3 * PCs with Oracle VM Virtual Box installed
- 1 * Test Manager
- 3 * Test Groups with 2 Testers in each group (See Annex A for details)
- GolfScore Program Version 1.1

6.0 Test Schedule

See Annex B for the full project schedule in Gantt Chart

Activity	Planned Start	Actual Start	Planned End	Actual End
Test Development	02-Oct-2017	02-Oct-2017	13-Oct-2017	13-Oct-2017
GolfScore V1.1 Availability	16-Oct-2017	23-Oct-2017	NA	NA
Test Setup	17-Oct-2017	24-Oct-2017	19-Oct-2017	25-Oct-2017
GolfScore Verification Test	19-Oct-2017	26-Oct-2017	27-Oct-2017	06-Nov-2017
GolfScore Regression Test	30-Oct-2017	07-Nov-2017	03-Nov-2017	10-Nov-2017
GolfScore Unscripted Test	06-Nov-2017	13-Nov-2017	10-Nov-2017	17-Nov-2017

7.0 Risks/Mitigation

Risk	Mitigation
Last minute requests leading to changes in requirements	- Test plan is planned and scheduled based on project timeline with 2 weeks of buffer - Any request for requirement changes to be handled by the Product Management Team and to be planned for subsequent releases
Unavailability of the assigned tester resources hinders the progress of the test	- A 2-men test team helps to cover the job of the other person when the person is down - Test Manager can step in to help as well if necessary
Occurrences of flaky tests affecting the progress of the test	- Re-run the test cases and proceed with the tests progress if the retest is a 'pass' - Log down the observations for post-mortem assessment to re-evaluate the situation after all the tests are done

8.0 Metrics

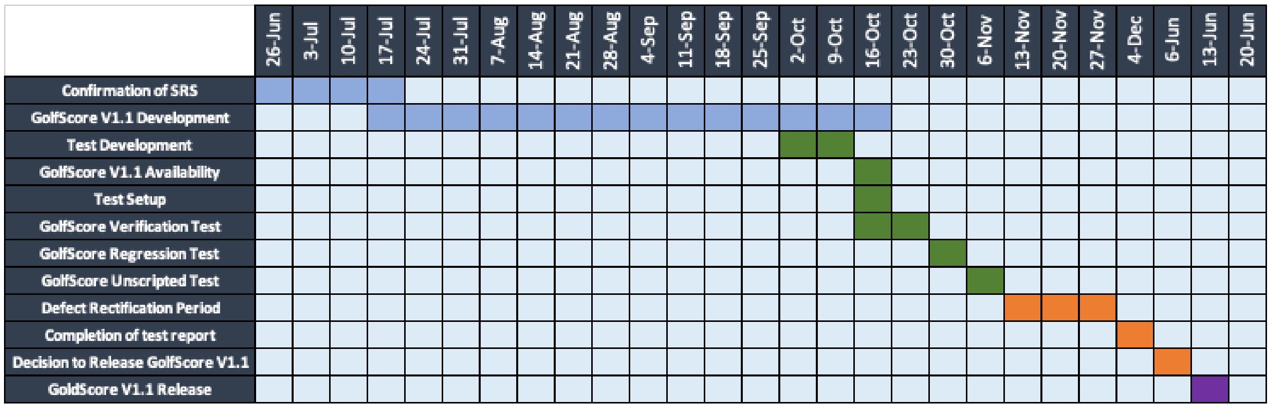
S/N	Metrics
1	Number of Test cases PASSED
2	Number of Test cases FAILED
3	Number of Test cases BLOCKED
4	Number of DEFECTS reported and filed
5	Total Number of Test Hours
6	Number of DEFECTS reported by USERS

Appendix A – Detailed Resource Requirements

Role	Personnel Name	Responsibilities
Test Manager	Michael Garner	- Reports test results to senior management - Provides technical guidance to the test team - Manages all required resources for the test activities
Group 1 Test Lead	Joshua Sloan	- Prepares and approves test report - Execute and supervise progress of assign test cases Group 1 - Execute regression test cases - Review all test results and reported defects - Provide technical opinions where needed
Group 1 Tester	Amy Maddox	- Execute assign test cases Group 1 - Log all test results - Report defects - Follow up on reported defects - Prepares test report
Group 2 Test Lead	Sarah Briggs	- Prepares and approves test report - Execute and supervise progress of assign test cases Group 2 - Execute regression test cases - Review all test results and reported defects - Provide technical opinions where needed
Group 2 Tester	Melissa Horne	- Execute assign test cases Group 2 - Log all test results - Report defects - Follow up on reported defects - Prepares test report
Group 3 Test Lead	Michelle Barlow	- Prepares and approves test report - Execute and supervise progress of assign test cases Group 3 - Execute regression test cases - Review all test results and reported defects - Provide technical opinions where needed
Group 3 Tester	Amanda Liew	- Execute assign test cases Group 3 - Log all test results - Report defects - Follow up on reported defects - Prepares test report

Appendix B – Detailed Test Schedule

The Gantt chart below shows the bigger project plan behind GolfScore Version 1.1.



	26-Jun	3-Jul	10-Jul	17-Jul	24-Jul	31-Jul	7-Aug	14-Aug	21-Aug	28-Aug	4-Sep	11-Sep	18-Sep	25-Sep	2-Oct	9-Oct	16-Oct	23-Oct	30-Oct	6-Nov	13-Nov	20-Nov	27-Nov	4-Dec	6-Jun	13-Jun	20-Jun
Confirmation of SRS	🟦	🟦	🟦	🟦																							
GolfScore V1.1 Development				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦										
Test Development															🟩	🟩											
GolfScore V1.1 Availability																	🟩										
Test Setup																	🟩										
GolfScore Verification Test																	🟩	🟩									
GolfScore Regression Test																			🟩								
GolfScore Unscripted Test																				🟩							
Defect Rectification Period																					🟧	🟧	🟧				
Completion of test report																								🟧			
Decision to Release GolfScore V1.1																									🟧		
GoldScore V1.1 Release																										🟪	

Appendix C – Test Cases

Test No	Test Case	Test Type
1	The program shall be able to load and run on PC with WINDOWS 2000	Non-Functional
2	The program shall be able to load and run on PC with WINDOWS XP	Non-Functional
3	The program shall be able to load and run on PC with WINDOWS Vista	Non-Functional
4	The program shall be able to load and run on PC with WINDOWS 7	Non-Functional
5	The program shall be able to load and run on PC with WINDOWS 8	Non-Functional
6	The program shall be able to load and run on PC with WINDOWS 10	Non-Functional
7	The program shall be able to load and run on PC with WINDOWS 11	Non-Functional
8	The program shall run with the command line option ‘-ctg’	Functional
9	The program shall run with the command line option ‘-c -t -g’	Functional
10	The program shall run with the command line option ‘-c’	Functional
11	The program shall run with the command line option ‘-t’	Functional
12	The program shall run with the command line option ‘-g’	Functional
13	The program shall run with the command line option ‘-ct’	Functional
14	The program shall run with the command line option ‘-cg’	Functional
15	The program shall NOT run with the command line option ‘-ca’ and an error message should be displayed	Functional
16	The program shall NOT run with the command line option ‘-a’ and an error message should be displayed	Functional
17	The program shall NOT run with the command line option ‘-cag’ and an error message should be displayed	Functional
18	The program shall display an error message when it is given an Invalid or non-existent Course Records	Functional
19	The program shall display an error message when it is given an Invalid or non-existent Golfer Records	Functional
20	The program shall run successfully with command line option ‘-ctg in.txt” where in.txt exists and is a valid file. Verify that three output files are produced after program has executed, namely “trank.rep”, “golfer.rep” and “course.rep”. If any of the 3 output files are already present, the user shall be prompted whether to overwrite.	Functional

Test No	Test Case	Test Type
21	The program shall run and exit with and error message with given the command line option '-ctg in.txt" where in.txt DOES NOT exists	Functional
22	The program shall run and exit with and error message with given the command line option '-ctg in.txt" where in.txt exists and is an INVALID file	Functional
23	The program accepts '1' as valid input for the number of golf course	Functional
24	The program accepts '5' as valid input for the number of golf course	Functional
25	The program does not accept '0' as valid input for the number of golf course	Functional
26	The program does not accept '6' as valid input for the number of golf course	Functional
27	The program does not accept '-1' as valid input for the number of golf course	Functional
28	The program does not accept 'A' as valid input for the number of golf course	Functional
29	The program accepts '1' as valid input for the number of golfers	Functional
30	The program accepts '12' as valid input for the number of golfers	Functional
31	The program does not accept '0' as valid input for the number of golfers	Functional
32	The program does not accept '13' as valid input for the number of golfers	Functional
33	The program does not accept '-1' as valid input for the number of golfers	Functional
34	The program accepts '3' as valid input for the par value	Functional
35	The program accepts '5' as valid input for the par value	Functional
36	The program does not accept '2' as valid input for the par value	Functional
37	The program does not accept '6' as valid input for the par value	Functional
38	The program does not accept '-3' as valid input for the par value	Functional
39	The program accepts '0' as valid input for the golfer score per hole	Functional
40	The program accepts '1' as valid input for the golfer score per hole	Functional
41	The program accepts '6' as valid input for the golfer score per hole	Functional
42	The program does not accept '7' as valid input for the golfer score per hole	Functional
43	The program does not accept '-5' as valid input for the golfer score per hole	Functional
44	The program does not accept any input that violates the delimiter constraints and shall	Functional

Test No	Test Case	Test Type
	return with an error message	
45	The program is being able to validate every golfer’s performance record [must have 18 holds, each with 3-5 strokes/per] from the in.txt file. If an invalid golfer’s record is found, the system shall output an appropriate error message	Functional
46	The program is able to calculate the golfer’s score for each golf course correctly	Functional
47	The program is able to calculate the golfer’s stroke count for each golf course correctly	Functional
48	The program is able to calculate the golfer’s score for the tournament correctly	Functional
49	The program shall run successfully with command line option ‘-c in.txt” where in.txt exists and is a valid file. The program shall output the “course.rep” file, and if an existing “course.rep” is already present, ask for permission to overwrite the file. Verify the format of course.rep is correct	Functional
50	The program shall run and exit with an error message with command line option ‘-c in.txt” where in.txt exists and is an INVALID file (does not contain course records or the records are not in correct format).	Functional
51	The program shall run successfully with command line option ‘-t in.txt” where in.txt exists and is a valid file. The program shall output the “trank.rep” file, and if an existing “trank.rep” is already present, ask for permission to overwrite the file. Verify the format of trank.rep is correct	Functional
53	The program shall run and exit with an error message with command line option ‘-t in.txt” where in.txt exists and is an INVALID file. (does not contain course & golfer records or the course & golfer records are not in correct format)	Functional
54	The program shall run successfully with command line option ‘-g in.txt” where in.txt exists and is a valid file. The program shall output the “golfer.rep” file, and if an existing “golfer.rep” is already present, ask for permission to overwrite the file. Verify the format of golfer.rep is correct	Functional
55	The program shall run and exit with an error message with command line option ‘-g in.txt” where in.txt exists and is an INVALID file. (does not contain golfer records or the golfer records are not in correct format)	Functional
56	The program shall run successfully **in less than 1 minute** with command line option ‘-ctg in.txt” where in.txt exists and is a valid file. Verify that three output files are produced after program has executed, namely “trank.rep”, “golfer.rep” and “course.rep”.	Non-Functional
57	The program shall run against the regression test cases and pass	Non-Functional