MSSE SOFTWARE, INC.

Test Plan for

(GolfScore 1.1)

Contents

1.0 Introduction

1.1. Objective

The Test Plan is an aggregation of information, which describes the entire test activity for this project. 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.

1.2. Project Description

The project GolfScore is a command line tool that will be used to generate scores of players in a golf course based on input score data. It will also help to produce reports based on this scores for the players participated in a game. It will also contain how a player performed and who won the tournament. It will run as a standalone executable program.

1.3. Process Tailoring

The project will be developed as per standard software development and testing life cycle; in this case we will use the **V** model of software development. The process includes requirements gathering and in accordance with it acceptance testing. Based on requirements the specifications are listed below:

1. Taking inputs from a text file containing well formatted data described in section 3.0.
2. Producing output with input data. Appendix B for more details about output data.
3. Command line execution of the application using built in commands also using an IDE.
4. Project will display specified error messages and either will stop or continue execution based on given input or output generation.

The SVT will be on the run alongside specifications listing. The project will be then go through the process of high level design as where to give input and what output will be from the inputs, the system integration testing will be performed.

The design of the various modules of the project will be tailored. That includes:

- Golf course settings
- Limits of player participation
- Calculating scores from par
- Assembling scores for a player in a tournament.
- Error handling when occurs.

Lastly the implementation in specified programming languages will be performed and it will follow the above processes to divide the entire projects into units and test them as individual units and also perform the overall integration testing when the units are working together.

Here we are omitting the full aspect of the development model as per our needs.

If we summarize, then the process will includes various phases of testing includes:

- Specification Testing
- System Integration Testing
- Unit Testing

Regression testing will be performed to attain the problems found in main testing to verify the integrity of the project and to confirm that the project is ready to deliverable.

1.4. Referenced Documents

1. Software Requirements Specification / Design Document, GolfSocre Version 1.1.

2.0 Assumptions/Dependencies

Here the assumptions timeframe for the testing listed above will be detailed in the section 6.0. In this section we will list the assumptions we gathered from the SRS.

1. The program will run on windows 2000 or later versions, so there will be compatibility issues.
2. It will be developed in C/C++.
3. The input file have to be formatted properly.
4. Input values should not alternate with garbage or unwanted values.
5. Command line application needs to use a language compiler to execute, hence it should be installed in the system.

The testing phase will assume to have available resources and tools to complete the integration, unit, SVT, regression testing.

3.0 Test Requirements

The application will run as a command line tool. Like the following example command:

- golf options filename output-directory

System Test requirements for the project includes but not limited to:

- Input Parameters
- Input Data tests
- Output Data tests.

Input Parameter tests:

1. Check the usable commands.
2. Check correct filename and correct output file format which is printable.
3. The program will stop giving incorrect command and will display correct error message.

4. If input file does not exist, then the input parameter error will be reported.
5. The following field of filename will be a directory, it is unavailable or not found, and then input parameter error will be reported.

Input Data tests:

1. Verify whether non-numeric data is given in place of numeric data will produce error message.
2. Par values should be 3, 4 or 5, otherwise program will pause with correct message.
3. Golfer with more than one record will be ignored except for first instance.

Output Data Tests:

1. Requested output report will be checked whether it was generated before.
2. User can choose by entering “Y” for rewriting the report again or “N” for discarded output.
3. If it’s not in the specified path, it will be created.

4.0 Test Tools

System test will require the following tools:

- CppUnit or Visual Studio test tools.
- An IDE like Visual Studio
- Input Data as text files
- The system codebase

5.0 Resource Requirements

For system dependencies, we will need:

- Windows 2000 or later version PC.
- MinGW or any other C/C++ compiler.

For generating output reports we will need:

- Input text file with correct formation. Input file formation is given in Appendix A.
- Enough memory to save and process the overall process.

The estimation for the project test is ~108 hours, encompassing the working hours only. Details of the test phases and roles are described in Appendix A.

6.0 Test Schedule

Based on the requirements and test planning the project shall be completed within the scheduled timeframe below:

Test Sequence	**Start**	**Finish**
Specification Testing	18-05-2021	23-05-2021
System Integration Testing	24-05-2021	27-05-2021
Unit Testing	28-05-2021	02-06-2021
Regression Testing	03-06-2021	05-06-2021

7.0 Test Cases

Based on the specifications and requirements, a list of test cases is given in Appendix B.

8.0 Risks/Mitigation

Potential risks that may have occur in this project and what to do to mitigate them are listed below:

1. Memory management: Development plan should consider handling memory leakage in the code base.
2. Optimization: Code execution should not take much memory or time to process.
3. Correct Output: Output should contain desired information, not unexpected symbols or data.
4. Compatibility issue

9.0 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

After shipment:

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

Size of software

10.0 Terms/Acronyms

SVT – System Verification Testing
DVT – Design Verification Testing
SRS – Software Requirement Specifications

Appendix A – Detailed Resource Requirements

Here's a detail outline of input text for the tests:

- Input: For generating reports we need input data as a text file, formation is given below:

1. Course Records:

   Column 1 – Blank

   Column 2-19 – Course Name

   Column 20 – Course Identifier

   Columns 21-38 – Par for holes 1-18; 3, 4, or 5.

2. Golfer Records:

   Column 1 – Blank

   Column 2 – Course Identifier

   Column 3-9 – Ignored

   Column 10-29 – Golfer's Name

   Column 30 – Ignored

   Column 31-48 – Stroke Count for each of the 18 holes.

The end of the input for course records will be determined by a blank delimiter record column and the golfer's one will be non-blank.

Project Task and Estimation:

Task	Members	Estimate efforts
Creating Test Specification	Test Designer	50 hours
Perform Test Execution	Tester	30 hours
Test Report	Tester	18 hours
Test Delivery	Tester, Test Administrator	20 hours
Total		**108 hours**

Appendix B – Test Cases

Test Serial	Description	Type
1	Command ‘golf -h’ should run without error	Functional
2	Command ‘golfscore’ should produce error	Functional
3	Command ‘golf –c input.txt output’ is correct	Functional
4	Command ‘golf –ctg input.txt output’ is correct	Functional
5	Command ‘golf –ctg input.txt output output2’ shall produce error	Functional
6	Par value ‘0’ , ‘1’ or ‘2’ shall produce error	Functional
7	Par value ‘3’ shall be accepted	Functional
8	Par value ‘5’ shall be accepted	Functional
9	Par value ‘5’ shall be accepted	Functional
10	The number of golf course '1' shall be accepted	Functional
11	The number of golf course '5' shall be accepted	Functional
12	The number of golf course '6' shall return an error	Functional
13	The number of golf course '0' shall return an error	Functional
14	Over par stroke will result to score 0	Functional
15	Par stroke will be score 1	Functional
16	Under par ‘1’ will add 2 to score	Functional
17	Under par ‘2’ will add 4 to score	Functional
18	Under par ‘3’ will add 6 to score	Functional
19	Number of golfers ‘2’ shall be accepted	Functional
20	Number of golfers ‘12’ shall be accepted	Functional
21	Number of golfers ‘8’ shall be accepted	Functional
22	Number of golfers ‘1’ shall return an error	Functional
23	Number of golfers ‘0’ shall return an error	Functional
24	Number of records ‘2’ for a course will be ignored except the first	Functional
25	Generated report will prompt ‘Y’/’N’ if entered again	Functional
26	Input file not in ‘.txt’ format shall produce error	Functional