MSSE SOFTWARE, INC.

Test Plan for
GolfScore

Contents

1. Introduction

1. Objective

This document describes the test plan for the GolfScore and includes information on what is to be tested, and how the testing is to be accomplished (test methodology). Specifically, this document describes the tests to be performed, the testing schedule, resources required, entry criteria, exit criteria, dependencies, test tools, metrics and the Test Plan Requirements Matrix. This is a living test plan and must be changed to reflect Core Team needs and requirements as they arise.

The main purpose of this test is to verify the requirements for the GolfScore as set forth in Software Requirements Specification/Design Document rev 1.1

2. Project Description

GolfScore is a program used to generate reports of golfers' results for a golf tournament. The input to the program will consist of a file containing two types of records. The output from the program will consist of up to 3 reports. The program is executed via a command line interface – there is no GUI associated with the application.

The program will be run as a stand-alone executable, and can be run from a command line prompt, from within an IDE (Integrated Development Environment), etc. Input to the program will come from an input record file, and output from the program will go to output record files in a format suitable for printing.

3. Process Tailoring

3.1 Description

According to the Software Requirements Specification/ Design Document GolfScore Revision 1.1,

System Verification Test will be broken into three phases:

Entrance Tests :

1. The program is Written in C or C++.
2. The program runs on a PC running on Windows 2000 or later versions.
3. The program can be run from command line prompt.
4. The program should run with valid parameters.

Main Functional Test will thoroughly verify the operation of the GolfScore software
Main Test determines that all the requirements in the SRS have been satisfied.
The Main Tests will consist of new tests written for this SW.
The following types of testing will be included:

➢ Specification testing
➢ Functional testing (Unit Testing, System Testing, UAT, Security Testing)
➢ Compatibility testing (Integration Testing)
➢ Limits testing

➢ Performance testing
➢ Documentation review
See Appendix C for a description of the Main Testing test cases.

Regression Test will be performed as a subset of Main Test after all problems found during Main Testing have been attended to. This means that all Severity 1 and 2 defects have been fixed and verified and that the product is ready for release.

3.2 References

1. Software Requirements Specification/ Design Document GolfScore Revision 1.1

2. Assumptions/Dependencies

In order to begin testing of the GolfScore, the following needs to happen:
- Approval of the proposed Test Plan by 05.10.24 to begin Requirements Verification testing
- Delivered functional code by 11.10.24 to begin Main Testing

3. Test Requirements

Test requirements extracted from the Software Requirements Specification/ Design Document GolfScore Revision 1.1:

- **Data Input.** Input to GolfScore should consist of a formatted text file containing the following records in the order given. Individual records in the file should be terminated by the end of a line.
  The name of the input file is supplied as a parameter on the program call line. [See Section 2.2. of a SRSD document]

- **Delimiter Record.** The end of the Course Records should be specified by a delimiter record with the following format:
  Column 1 Non-blank

  The end of the Golfer Records, and hence of the input file itself, will also be specified by a delimiter record:
  Column 1 Non-blank

- **Golfer Records.** There should be one Golfer Record for each golfer for each course played; these records can appear in any order. Each record should contain the name of the golfer, the Course Identifier, and the golfer's stroke count on each of the 18 holes:
  Column 1 Blank
  Column 2-19 Course name
  Column 20 Single-character course identifier
  Column 21-38 Par for holes 1-18, in order, single integer: 3, 4 or 5

- **Data output.** GolfScore should generate up to 3 reports, based on input options. The generated reports should be stored as text files in the directory determined from the program call line.

- **Tournament Ranking Report.** A list of all the golfers with the golfer's name, the score for each course, the total tournament score, and that golfer's final standing (1st place, 2nd place, etc.). The list should be in descending order of final score (i.e. best golfer first). In the case of ties, the golfers will be listed alphabetically.

The output filename for this report will be *trank.rep*

- **Golfer Report.** This report is exactly the same as the Tournament Ranking Report except that the golfers should be listed alphabetically by last name.
  The output filename for this report will be *golfer.rep*

- **Course Report.** This report will have one section for each Golf Course specified in the input Course Records. For each course, the report will show a list of all the golfers with the golfer's name, the hole-by-hole stroke count for that course, and the total score for that course, listed in descending order of score on that course.
  The output filename for this report will be *course.rep*

- **Input Parameter Errors.** For any input parameter errors, including unrecognisable options, the program should stop, and display a message explaining the error.
  The first field following *options* should be assumed to be a filename. If the file specified by *filename* does not exist, an input parameter error should be reported.
  The first field following *filename* should be assumed to be a directory. If the directory specified by *output-directory* does not exist, an input parameter error should be reported. If *output-directory* is not supplied, the directory that contains *filename* should be used.
  Any field beyond the *output-directory* field should be ignored and not considered an error.

- **Input Data Errors.** Input Data errors checked for are as follows:
  i) Non-numeric data where numeric data is expected: the program should stop with an appropriate error message
  ii) Par values that are not 3, 4, or 5: the program should stop with an appropriate error message.
  iii) Any golfer that has two or more records for the same golf course: the additional records after the first should be ignored, a message should be displayed, and processing to continue.

- **Errors on Output.** If any of the requested output reports already exists, the program should pause and ask the user if the file should be overwritten. Sample prompt: "*File <file> already exists. Do you want to overwrite it? (Y/N)*".
  The user should then respond "Y" or "N". If "Y", the output file should be overwritten; if "N", the generated output should be discarded. There should be a separate user prompt for each output report type requested.
  If the specified output report does not exist in the path specified, it should be created.
  If any other errors occur while attempting to write the output files, program behaviour is unspecified.

4. Test Tools

Apart from the deliverable product following software test tools are used to assist in the testing process:

- Issue & Project Tracking Software (Jira)
- Installation media for multiple Windows version above 2000.

5. Resource Requirements

System Verification Test will require the following resources:

- A GolfScore SW dedicated to Advanced testing

- Four Software Verification and Testing Track test personnel with 100% of his/her time available for this effort.
- Two Software Development team personnel with 100% of his/her time available for this effort.
- A virtualisation software

Total Estimated Hours: 360 hours
For a detailed Resource Requirements review please see Appendix A

6. Test Schedule

The testing defined in this document shall be completed according to the following schedule:

Test Sequence	Start	Finish
1. Test Development	02.10.24	03.10.24
2. Main Testing	05.10.24	19.10.24
3. SW Regression Testing	17.10.24	20.10.24
4. Defect management	20.10.24	25.10.24

For more detailed Gannt chart view and PERT diagram please study Appendix B

7. Risks/Mitigation

Risks which could affect the Test Plan development and realisation listed below:

- Failed specification testing - delays the start of the development, which affects all the following steps
- Lack of Test/Development manpower - possible delay
- Disapproval of the suggested test plan by a higher management for any reason - delay
- Change of Customer's Specification Requirements later during the SDLC - complete change of the SDLC Plan and a delay

8. Metrics

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

The status and progress of SVT will be recorded through the collection of various sets of data, and the Test Case Matrices will regularly be updated with the status of each test case. Thus, at any time one can see how many test cases have been attempted and, of those, how many have passed.
In addition, effort, size and defect data will be collected prior to and after product shipment. Once data from enough projects has been collected, estimates of testing progress and duration will become more meaningful.

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

Appendix A – Detailed Resource Requirements

1. **Test Activity Breakdown**
List all the key test activities involved in the project:

- **Test Planning**: Writing the test plan, including scope, objectives, resources, schedule, and risk management.
- **Test Case Design:** Creating detailed test cases based on requirements, design documents, and user stories.
- **Test Environment Setup:** Configuring test environments
- **Test Data Preparation:** Creating and setting up test data required for executing the test cases.
- **Unit Testing:** Testing individual components or modules of the application.
- **Integration Testing:** Testing combined components to ensure they work together as expected.
- **System Testing:** Testing the entire system for functionality, performance, and security.
- **User Acceptance Testing (UAT):** Coordinating with users to validate the system meets business requirements.
- **Regression Testing:** Testing existing functionality after new changes are introduced.
- **Performance Testing:** Testing the system's performance under various conditions.
- **Security Testing:** Testing the system for vulnerabilities and ensuring it is secure from threats.
- **Defect Management:** Logging, tracking, and retesting defects found during testing.
- **Test Reporting:** Preparing and distributing test execution reports, defect reports, and summary reports.
- **Test Review and Retrospective:** Reviewing the testing process to identify areas for improvement.

2. **Estimating Hours for Each Activity**

Test Activity	Estimated Hours	Responsible Test Engineer
Test Planning	20	[Test Engineer 1]
Test Case Design	50	[Test Engineer 2]
Test Environment Setup	15	[Test Engineer 3]
Test Data Preparation	10	[Test Engineer 4]
Unit Testing	40	[Test Engineer 1]
Integration Testing	30	[Test Engineer 2]
System Testing	50	[Test Engineer 3]
User Acceptance Testing (UAT)	20	[Test Engineer 4]

Regression Testing	25	[Test Engineer 1]
Performance Testing	30	[Test Engineer 2]
Security Testing	25	[Test Engineer 3]
Defect Management	20	[Test Engineer 4]
Test Reporting	15	[Test Engineer 1]
Test Review and Retrospective	10	[Test Engineer 2]
Total Estimated Hours	**360**	

3. Assigning Resources

- Assign specific test engineers to each activity if possible. This helps in better tracking and accountability.
- Consider the skill set of each test engineer to match them with the appropriate tasks.

4. Grand Total of Effort

Total Estimated Hours: 360 hours

5. Considerations for Estimations

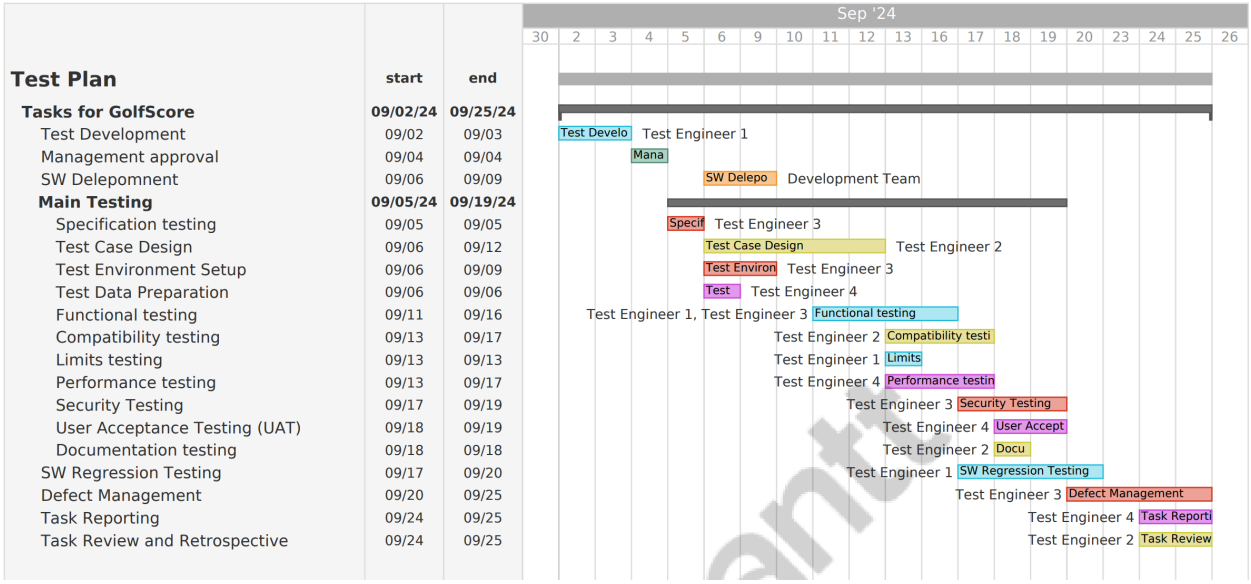
- **Buffer Time:** Include buffer time to account for unforeseen challenges or delays.
- **Team Availability:** Consider the availability of each test engineer, including holidays, vacations, and other commitments.
- **Parallel Activities:** Identify activities that can be done in parallel to optimise resource usage.

6. Adjusting Estimates

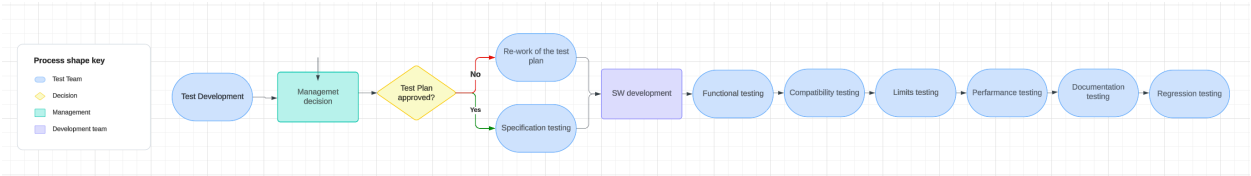
- Review the estimates with the test team and stakeholders for accuracy.
- Adjust based on feedback, risks, or changes in project scope.

Appendix B – Detailed Test Schedule

Test Plan Gannt chart view:



… and PERT diagram:



Appendix C – Detailed Test Cases

1	The number of golf course '1' shall be accepted	Functional
2	The number of golf course '5' shall be accepted	Functional
3	The number of golf course '6' shall return an error	Functional
4	The number of golf course '0' shall return an error	Functional
5	The number of golfers ‘0' shall return an error	Functional
6	The number of golfers ‘1’ shall return an error	Functional
7	The number of golfers ’13' shall return an error	Functional
8	The number of golfers ’8’ shall be accepted	Functional
9	The par value of ’8’ shall return an error	Functional
10	The par value of ’1’ shall return an error	Functional
11	The par value of ’3’ shall be accepted	Functional
12	If the input file specified by *filename* does not exist, an input parameter error shall be reported.	Functional
13	If the directory specified by *output-directory* does not exist, an input parameter error shall be reported	Functional
14	The number of golf course ‘abc' shall return an error	Functional
15	The golfer’s score ’18’ shall be accepted	Functional
16	The golfers score ’18 22’ shall be accepted, with a specific message explaining that first results is being ignored	Functional
17	The program shall be Written in C or C++.	Non-functional
18	The program shall run on a PC running windows 2000	Non-functional
19	The program shall run on PC running windows 7	Non-functional
20	The program shall run on PC running windows 8	Non-functional
21	The program shall run on PC running windows 10	Non-functional
22	The program shall run on PC running windows 11	Non-functional
23	The program shall run as a stand-alone executable	Non-functional