MSSE SOFTWARE, INC.

**Test Plan for**

**GolfScore**



Contents

**1.0 Introduction [3](#_8ndp5oufdw7q)**

**1.1. Objective [3](#_a8f7jznm04hz)**

**1.2. Project Description [3](#_ct75r4rociaz)**

**1.3. Process Tailoring [3](#_r0fqeeszw09l)**

**1.4. Referenced Documents [3](#_o17ow8ksdp1b)**

**2.0 Assumptions/Dependencies [3](#_c42pzuw7wdch)**

**3.0 Test Requirements 4**

**4.0 Test Tools [4](#_71xp59knv9kw)**

**5.0 Resource Requirements 5**

**6.0 Test Schedule 5**

**7.0 Risks/Mitigation 5**

**8.0 Metrics 5**

**Appendix A – Detailed Resource Requirements 6**

**Appendix B – Detailed Test Schedule 6**

**Appendix C– Test Cases 7**

## **1.0 Introduction**

### **1.1 Objective**

The Test Plan is an aggregation of information which describes the entire testing activity for this project. It encompasses all levels of testing, including unit, development testing, system verification testing (SVT), and beta testing. It defines product requirements, testing scope, schedule, resource allocation, quality goals, assumptions, exclusions, and risk assessments.

This test plan outlines the testing strategy for GolfScore Release 1.1, a command-line software application developed to process golf tournament scores and produce structured output reports. The plan provides the testing objectives, tools, resource needs, risk considerations, and schedules to ensure the software fulfills its functional and performance requirements.

### **1.2 Project Description**

GolfScore is a command-line program designed to process input files containing golf tournament data. It generates three types of reports:

* Tournament Ranking Report
* Golfer Report
* Course Report

The application supports a maximum of 5 courses and 12 golfers, and the outputs are stored as plain-text files intended for printing.

### **1.3 Process Tailoring**

This project follows a tailored software testing lifecycle based on best practices in specification testing, functional testing, limit/boundary testing, performance testing, and error-handling validation. Testing will include both manual and automated approaches. Certain documentation phases are condensed due to the project’s scope, and GUI testing is excluded as GolfScore is CLI-based.

### **1.4 Referenced Documents**

* GolfScore Software Requirements Specification (SRS), Rev. 1.1, July 18, 2017
* Test Plan Template
* Test Plan Example

## **2.0 Assumptions/Dependencies**

### **Assumptions:**

* The input file format adheres strictly to the structure defined in the SRS.
* The software will be executed on Windows 2000 or later systems.
* The scoring system and report formats are fixed as defined in the SRS.

### **Dependencies:**

* Delivery of the final executable for GolfScore prior to test execution.
* Test data files formatted according to SRS specifications.
* Development support is available to resolve critical issues during testing.

## **3.0 Test Requirements**

### **Functional Requirements:**

* Validate correct parsing of Course Records, Delimiters, and Golfer Records.
* Ensure score calculation matches the defined scoring rules.
* Generate Tournament Ranking, Golfer, and Course Reports correctly.
* Handle malformed input files and incorrect data values gracefully.
* Prompt before overwriting output files if they already exist.

### **Non-Functional Requirements:**

* Ensure performance requirements are met (max execution time: 1 minute for full dataset).
* Validate compatibility with specified OS versions.
* Confirm robustness of error handling and resilience to invalid inputs.

## **4.0 Test Tools**

* Text Editor (Notepad++ / VS Code): Manual output validation.
* Command Line Terminal: Execution environment.
* Custom Batch/Shell Scripts: Automation of command-line test execution.
* Logging Tools: For capturing outputs and errors.
* Excel/TestRail: Managing test cases and recording results.
* Diff Tools (WinMerge / Beyond Compare): Comparing expected vs actual reports.

## **5.0 Resource Requirements**

| **Resource** | **Quantity** | **Description** |
| --- | --- | --- |
| Test Engineers | 2 | Responsible for test design and execution |
| Windows PC | 1 | Used for test execution and report validation |
| Test Data Files | N/A | Includes valid and invalid input test cases |
| Developer Support | 1 | For debugging and resolving test defects |

(Refer to Appendix A for detailed breakdown)

## **6.0 Test Schedule**

| **Task** | **Start Date** | **End Date** |
| --- | --- | --- |
| Test Preparation | Day 1 | Day 2 |
| Functional Testing | Day 3 | Day 7 |
| Error Handling Testing | Day 8 | Day 9 |
| Performance Testing | Day 10 | Day 10 |
| Documentation & Reporting | Day 11 | Day 11 |

(Refer to Appendix B)

## **7.0 Risks/Mitigation**

| **Risk** | **Impact** | **Mitigation Strategy** |
| --- | --- | --- |
| Incorrectly formatted input test files | High | Provide verified templates and automated validators |
| Limited QA manpower | Medium | Automate repetitive tests and prioritize critical cases |
| Late delivery of executable | High | Add buffer days before test execution phase |
| Unexpected behavior during file I/O | Medium | Mock edge cases and include logging for all I/O events |

## **8.0 Metrics**

### **Prior to Shipment:**

* Effort hours for DVT, SVT, and Regression
* of defects found per phase
* Test tracking S-Curve (test execution vs time)
* PTR (Problem Tracking Report) S-Curve

### **After Shipment:**

* Defects found in production and their mapped dev phase
* Size of the final software (LOC, # of modules/files)

## **Appendix A – Detailed Resource Requirements**

| **Activity** | **Resource** | **Hours** |
| --- | --- | --- |
| Test Plan Creation | QA Engineer 1 | 8 |
| Test Case Development | QA Engineer 2 | 12 |
| Input File Preparation | QA Engineers | 10 |
| Manual Test Execution | QA Engineers | 24 |
| Regression Testing | QA Engineer 1 | 10 |
| Final Report & Metrics Logging | QA Engineer 2 | 6 |
| **Total** | - | **70** |

## **Appendix B – Detailed Test Schedule**

### 1. Preparation (Start → Preparation)

### - Time: 2 days

### - Status: Done

### 2. Functional Testing (Preparation → Functional)

### - Time: 3 days

### - Depends on: Preparation

### 3. Error Handling (Functional → Error Handling)

### - Time: 2 days

### - Depends on: Functional

### 4. Performance Testing (Error Handling → Performance)

### - Time: 3.3 days

### - Depends on: Error Handling

### 5. Report & Closure (Performance → Report → End)

### - Time: 1.5 days

### - Depends on: Performance

## **Appendix C – Test Cases**

**Test Case 1: Validate Course Record Parsing**

* Input: Valid course record
* Expected Output: Parsed name, ID, and par values for 18 holes

**Test Case 2: Validate Golfer Record Parsing**

* Input: Valid golfer record
* Expected Output: Correct parsing of name, ID, strokes for 18 holes

**Test Case 3: Tournament Ranking Report**

* Input: 3 courses, 6 golfers
* Expected Output: Tournament ranking in descending order of scores

**Test Case 4: Input Data Error Handling**

* Input: File with non-numeric stroke count
* Expected Output: Error message; program halts

**Test Case 5: Performance Validation**

* Input: 5 courses, 12 golfers
* Expected Output: Processing completes within 1 minute

**Test Case 6: Combined Report Generation**

* Input: CLI options -ctg with full input
* Expected Output: trank.rep, golfer.rep, course.rep with valid content

**Test Case 7: File Overwrite Prompt**

* Input: Output file already exists
* Expected Output: Prompt to overwrite, handle "Y/N" input correctly