TEST IT, INC.®️

**Test Plan for**

**GolfScore**

**Revision 1.1**

**July, 2023**

**Sijan Dev**

Contents

1.0 Introduction 3

1.1. Objective 3

1.2. Project Description 3

1.3. Scope of Testing 3

1.4. Test Description 3

1.5. Referenced Documents 4

2.0 Assumptions/Dependencies 4

3.0 Test Requirements 5

4.0 Test Tools 6

5.0 Resource Requirements 7

6.0 Test Schedule 8

7.0 Risks/Mitigation 8

8.0 Metrics 9

Appendix A – Detailed Resource Requirements 11

Appendix B – Detailed Test Schedule 12

Appendix C – Brief Test Cases 13

# Introduction

## Objective

The purpose of the Test Plan for the GolfScore software (Release 1.1) is to provide a clear and structured approach to testing. It outlines the scope, objectives, and resources required for the testing process. The Test Plan aims to ensure the software's functionality, accuracy, and reliability while identifying potential risks and issues. It assigns roles and responsibilities to team members and sets a timeline for testing activities. The Test Plan also documents testing procedures, ensuring thorough test coverage and providing a roadmap for delivering a high-quality and reliable software product to end-users.

## Project Description

GolfScore is a stand-alone software program designed for generating reports of golfers' results in a golf tournament. It operates via a command-line interface, processing input data from a formatted text file. The software generates up to three types of reports: the Course Report, the Tournament Ranking Report, and the Golfer Report. GolfScore accurately calculates golfer scores based on the number of strokes under or over par for each hole. With support for 1 to 5 golf courses and 2 to 12 golfers, GolfScore provides a reliable and efficient solution for golf tournament result analysis.

## Scope of Testing

The scope of testing for GolfScore includes thoroughly evaluating all aspects of the software's functionality and features to ensure it meets the specified requirements. Testing will cover the command-line interface, input file handling, report generation accuracy, scoring calculations, error handling, performance, compatibility, usability, security, data validation, regression, and documentation. The goal is to deliver a reliable and efficient software solution for generating golf tournament reports.

## Test Description

GolfScore is a program used to generate tournament results for golfers on each course. This program takes one input file and produces three output text files.

Following are the test phases:

* Entrance Tests

The Entrance Test Phase verifies requirements related to initial setup, input handling, and error handling.

* Main Tests

The Main Test Phase verifies requirements for generating different types of reports, scoring calculations, and data accuracy.

* Exit Tests

The Exit Test Phase verifies requirement related to handling multiple records for the same golfer and displaying the appropriate message.

* Regression Tests

After code changes & bug fixes, all tests will be re-run to ensure proper system behavior.

## Referenced Documents

The referenced documents are:

1. GolfScore (Release 1.1) Software Requirements Specifications (SRS) Document.
2. System Verification Test Plan for Advanced Color Module, Revision 2.

# Assumptions/Dependencies

* Requirements Assumptions

It is assumed that the Software Requirements Specification (SRS) document is complete, accurate, and covers all necessary functionalities. Any changes or clarifications to requirements will be communicated promptly to the testing team.

* Test Scope Assumptions

The test scope is assumed to be based on the SRS document, and any additional requirements will be communicated and agreed upon before proceeding with testing.

* Unit Testing Responsibility

The development team is assumed to conduct thorough unit testing for individual modules to ensure their correctness and functionality before handing them over for integration testing.

* Integration Testing Responsibility

The integration testing responsibility lies with the testing team, who will ensure that the individual modules work harmoniously when integrated into the complete GolfScore software.

* SVT Responsibility

The testing team will perform System Verification Testing to validate the entire system and verify its compliance with the specified requirements.

* Regression Testing Responsibility

The testing team will be responsible for conducting regression testing to ensure that any new code changes or bug fixes do not adversely impact previously tested functionalities.

* Beta Testing Responsibility

The testing team will coordinate and execute beta testing with end-users to gather real-world feedback and ensure the software performs as expected in the target environment.

* Code Completion Dependency

The testing schedule assumes that the development team will complete coding for all planned features and modules within the specified timeline, allowing sufficient time for testing activities.

* Prototype Availability

It is assumed that a functional prototype of GolfScore will be available to the testing team by a specified date to initiate the testing process.

* Test Environment Availability

The required test environment, including hardware, software, and test data, will be available and set up before testing commences.

* Timely Bug Fixes

The development team is expected to address and resolve reported bugs and issues in a timely manner, allowing the testing team to retest and verify the fixes promptly.

* Collaboration & Communication

Assumption of effective collaboration and communication between the development and testing teams to share updates, address issues, and ensure a smooth testing process.

# Test Requirements

1. Entrance Tests
2. The software must support a command-line interface (CLI) for user interaction.
3. The CLI must accept valid options, such as -h, -c, -t, -g, and their combinations (e.g., -cg).
4. The software must handle input parameter errors, such as unrecognizable options, and display appropriate error messages.
5. If the input file specified by the user does not exist, the software must report an input parameter error.
6. If the output directory specified by the user does not exist, the software must report an input parameter error.
7. The software must handle non-numeric data and par values that are not 3, 4, or 5 with appropriate error messages.
8. The software must prompt the user for confirmation before overwriting any existing output files.
9. Main Tests
10. The software must generate the Course Report (-c) when the corresponding option is selected.
11. The Course Report must display the list of golf courses, including the name, course identifier, and par for each hole.
12. The software must generate the Tournament Ranking Report (-t) when the corresponding option is selected.
13. The Tournament Ranking Report must display a list of all golfers with their names, scores for each course played, total tournament score, and final standing.
14. The golfers in the Tournament Ranking Report must be listed in descending order of their final scores. In case of ties, golfers should be listed alphabetically.
15. The software must generate the Golfer Report (-g) when the corresponding option is selected.
16. The Golfer Report must display a list of all golfers in alphabetical order by last name, along with their scores for each course played.
17. The software must correctly calculate each golfer's score for each hole based on the number of strokes under or over par.
18. Each golf course must have 18 holes, and the par for each hole must be 3, 4, or 5 strokes.
19. A golfer's total stroke count for a particular golf course must be the sum of the stroke counts for each of the 18 holes.
20. A golfer's total score for a particular golf course must be the sum of the scores for each of the 18 holes.
21. A golfer's total tournament score must be the sum of his or her scores for all courses played.
22. Exit Tests
23. The software must handle any golfer with multiple records for the same golf course by ignoring additional records after the first and displaying a message.
24. Other Miscellaneous Tests
25. Validate that GolfScore runs smoothly on a PC running Windows 2000 or any later version.
26. Verify that GolfScore completes its processing within one minute, meeting the specified performance requirement.

# Test Tools

1. Test Management & Issue Tracking Tool

A widely-used test management tool like Jira will be used to manage test cases, track defects, and monitor testing progress.

1. Test Automation Tool

Selenium or similar automation frameworks will be utilized to automate repetitive test cases and ensure consistent test execution.

1. Performance Testing Tool

JMeter will be used to conduct performance testing and measure the system's response under different load conditions.

1. Version Control System

Git will be used to manage test scripts, test data, and test documentation, ensuring version control and collaboration among team members.

1. Code Coverage Tool

A code coverage tool, like JaCoCo will be employed to measure the code coverage achieved by automated tests.

1. CI Tool

Jenkins will be used for automating the build, deployment, and test execution processes.

1. Test Environment Management Tool

A configuration management tool like Docker will be used to manage test environments, ensuring consistency and reproducibility.

1. Other Miscellaneous Tool

Installation media for multiple Windows version above 2000.

# Resource Requirements

1. Test Team

Test Manager: 1

Test Analysts: 2

Test Automation Engineers: 2

Domain Experts: 1

Test Environment Manager: 1

Test Tool Experts: 1

1. Development Team Collaboration

Collaboration with 2 developers from the development team for bug triaging, defect verification, and bug fixes.

1. Hardware Resources

Test Machines: 5

Server Machine: 1

Mobile Devices for Compatibility Testing: 3 (Android, iOS)

Printers and Printing Devices: 2

Scanners (Optional): 1

1. Test Data Management

1 resource responsible for creating and managing test data.

1. Documentation Team

1 documentation expert to maintain test documentation.

1. Training & Skill Development

Investment in training for the test team.

1. Test Tool Development Team

2 developers for building and maintaining custom test tools.

1. Project Management & Coordination

1 project manager and 1 coordinator.

1. Infrastructure & Network Support

IT support for managing network infrastructure and hardware availability.

1. Time & Effort

Full-time commitment from all team members throughout the testing lifecycle.

1. Budget

Allocated budget for testing activities, including tool licensing, hardware procurement, and resource training.

1. User Acceptance Testing Participants

5 end-users or representatives for UAT.

1. Stakeholder Support

Cooperation and support from stakeholders for smooth test execution and issue resolution.

# Test Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **ACTIVITY** | **START DATE** | **END DATE** | **DURATION (Days)** |
| **Test Planning & Creation** | 2023-08-01 | 2023-08-05 | 5 |
| **Test Environment Setup** | 2023-08-06 | 2023-08-10 | 5 |
| **Test Case Design & Review** | 2023-08-11 | 2023-08-20 | 10 |
| **Entrance Test Exec.** | 2023-08-21 | 2023-08-25 | 5 |
| **Entrance Test Report Prep.** | 2023-08-26 | 2023-08-30 | 5 |
| **Main Test Exec.** | 2023-08-31 | 2023-09-14 | 15 |
| **Main Test Report Prep.** | 2023-09-15 | 2023-09-24 | 10 |
| **Regression Test Exec.** | 2023-09-25 | 2023-09-30 | 6 |
| **Regression Test Report Prep.** | 2023-10-01 | 2023-10-05 | 5 |
| **UAT Exec.** | 2023-10-06 | 2023-10-15 | 10 |
| **Defect Triage & Verification** | 2023-10-16 | 2023-10-25 | 10 |
| **Final Test Report Prep.** | 2023-10-26 | 2023-10-31 | 6 |
| **Test Closure & Handover** | 2023-11-01 | 2023-11-05 | 5 |

# Risks/Mitigation

1. Incomplete or Ambiguous Requirements

**Risk:** The Software Requirements Specification/Design Document may have incomplete or ambiguous requirements, leading to uncertainty in test case design and testing coverage.

**Mitigation:** Conduct thorough reviews of the SRS document and seek clarifications from stakeholders. Collaborate closely with the development team to ensure a shared understanding of requirements.

1. Test Environment Readiness

**Risk:** Delays in setting up the required test environment (hardware, software, and data) could hinder the test execution process.

**Mitigation:** Start test environment setup early in the testing phase. Plan and coordinate with the IT team to ensure all necessary components are available on time.

1. Resource Constraints

**Risk:** Insufficient availability of skilled testers, domain experts, and tool experts can impact the testing timeline and quality.

**Mitigation:** Properly assess the resource requirements and allocate skilled personnel to the project. Conduct training sessions if necessary to bridge any skill gaps.

1. Time Constraints

**Risk:** Unexpected delays in development, changes in project schedules, or tight deadlines may compress the testing timeline.

**Mitigation:** Continuously monitor the project progress and communicate any deviations from the plan to the project management team. If needed, prioritize testing efforts and focus on critical areas**.**

1. Defect Resolution Delays

**Risk:** Prolonged time to resolve defects found during testing can impact the overall testing progress and delivery schedule.

**Mitigation:** Implement a well-defined defect tracking and resolution process. Collaborate closely with the development team to expedite defect fixes and verify them promptly.

1. Integration Issues

**Risk:** Integration of GolfScore with other systems or external components may result in compatibility or data exchange issues.

**Mitigation:** Conduct thorough integration testing, including data interchange scenarios, to identify and address integration-related problems early in the testing phase.

1. Scope Creep

**Risk:** Changes in requirements or scope during the testing phase can disrupt planned test efforts.

**Mitigation:** Establish a clear change management process, and involve stakeholders in evaluating the impact of scope changes on testing. Update the test plan and schedule accordingly.

1. User Acceptance Testing Challenges

**Risk:** Challenges in coordinating User Acceptance Testing (UAT) with end-users and obtaining timely feedback may delay the final release.

**Mitigation:** Proactively plan and communicate the UAT schedule with end-users. Provide necessary support and conduct regular meetings to gather feedback promptly.

# 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

Appendix A – Detailed Resource Requirements

|  |  |  |
| --- | --- | --- |
| **ACTIVITY** | **REPONSIBLE QA** | **EFFORT (HOURS)** |
| **Entrance Testing** |  |  |
| **Test Case Design** | Test Manager | 40 |
| **Test Environment Setup** | Test Environment Manager | 20 |
| **Test Exec.** | Test Automation Er 1 | 60 |
| **Test Analysis & Report Prep.** | Test Analyst 1 | 20 |
| **Main Testing** |  |  |
| **Test Case Review** | Test Analyst 2 | 10 |
| **Main Test Exec.** | Test Automation Er 1, 2 | 100 |
| **Defect Reporting** | Test Analyst 1 | 30 |
| **Defect Verification** | Test Analyst 1 | 25 |
| **Test Analysis & Report Prep.** | Test Analyst 2 | 20 |
| **Exit Testing** |  |  |
| **Exit Criteria Review** | Test Manager | 10 |
| **Test Closure Report** | Test Analyst 1 | 20 |
| **Lessons Learned** | Test Manager | 15 |
| **Regression Testing** |  |  |
| **Test Case Selection** | Test Analyst 2 | 15 |
| **Regression Test Exec.** | Test Automation Er 2 | 75 |
| **Defect Reporting** | Test Analyst 1 | 15 |
| **Defect Verification** | Test Analyst 1 | 10 |
| **Test Analysis & Report Prep.** | Test Analyst 2 | 15 |
| **Documentation Review** |  |  |
| **Document Review** | Test Analyst 1 | 25 |
| **Total Effort** | | **525** |

Appendix B – Detailed Test Schedule

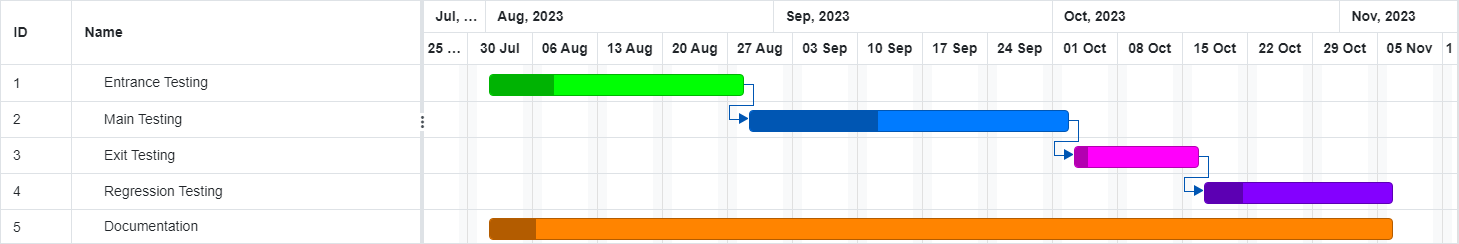


Fig B.1 : Gnatt Chart

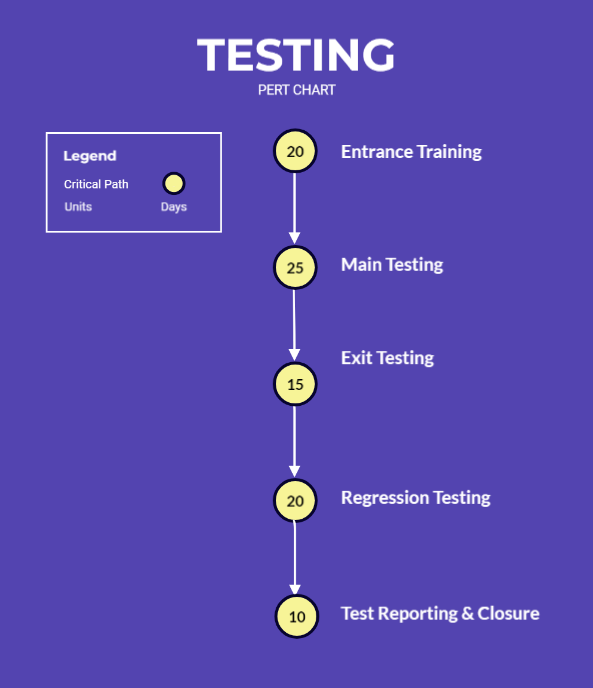


Fig B.2 : PERT Chart

Appendix C – Brief Test Cases

|  |  |  |
| --- | --- | --- |
| **No.** | **Test Case** | **Test Type** |
| **1** | The program shall be Written in C or C++. | Non-functional |
| **2** | The program shall run on a PC running windows 2000 | Non-functional |
| **3** | The program shall run on PC running windows XP | Non-functional |
| **4** | The program shall run on PC running windows VISTA | Non-functional |
| **5** | The program shall run on PC running windows 7 | Non-functional |
| **6** | The program shall run on PC running windows 8 | Non-functional |
| **7** | The program shall run on PC running windows 10 | Non-functional |
| **8** | The program shall run as a stand-alone executable | Non-functional |
| **9** | Command line options -ctg shall be accepted | Functional |
| **10** | Command line options –c shall be accepted | Functional |
| **11** | Command line options –t shall be accepted | Functional |
| **12** | Command line options –g shall be accepted | Functional |
| **13** | The number of golf course 1 shall be accepted | Functional |
| **14** | The number of golf course 5 shall be accepted | Functional |
| **15** | The number of golf course -5 shall be accepted | Functional |
| **16** | The number of golf course 6 shall be accepted | Functional |
| **17** | The number of golf course 0 shall return an error | Functional |
| **18** | The number of golfers 1 shall return an error | Functional |
| **19** | The number of golfers 2 shall be accepted | Functional |
| **20** | The number of golfers 12 shall be accepted | Functional |
| **21** | The number of golfers 13 shall return an error | Functional |
| **22** | Par for hole 2 shall return error | Functional |
| **23** | Par for hole 6 shall return error | Functional |
| **24** | Par for hole 3 shall be accepted | Functional |
| **25** | Par for hole 4 shall be accepted | Functional |
| **26** | Par for hole 5 shall be accepted | Functional |
| **27** | Calling the program with command line option –ctg shall generate 3 output file trank.rep, golfer.rep, course.rep. If any of the file already exit the user shall prompted with a message that say file already exits and asking it to overwrite it or not. | Functional |
| **28** | Calling the program with command line option –c shall generate output file course.rep. If file already exit the user shall prompted with a message that say file already exits and asking it to overwrite it or not. | Functional |
| **29** | Calling the program with command line option –t shall generate output file trank.rep. If file already exit the user shall prompted with a message that say file already exits and asking it to overwrite it or not. | Functional |
| **30** | Calling the program with command line option –g shall generate output file golfer.rep. If file already exit the user shall prompted with a message that say file already exits and asking it to overwrite it or not. | Functional |
| **31** | If output cannot be saved due to insufficient permission the program shall display error. | Functional |