**Ten-Pin Bowling Game Backend Testing**

**Summary Report**

1. **Report Identifier**

Bowling Game Backend Unit Testing

GitHub Link : <https://github.com/Havenhemant/GameTesting/>

1. **Introduction**

There are nine-unit tests created in the Python file, and a refactoring of the provided game code implementation. And test whether the game meets all business requirements for scoring calculation. The following key functionalities are tested in the report.

1. First test created for normal frame scoring without strikes or spares score
2. There is also a performed strike detection and bonus calculation
3. Spare detection and bonus calculation performed in unit test
4. Tenth frame result testing with special rules handling
5. Perfect game scenario unit test with 300 points
6. Input Validation and error handling check, but not available and implemented in the code.
7. There is also a gutter game unit test performed for where all rolls knock down zero pins.
8. Unit test where all rolls knock down exactly one pin.
9. **Purpose**

**The purpose** of this testing was to validate whether the bowling game scoring functions executed and performed correctly or not. The correctness of the score result is tested by the nine unit tests. The testing ensures that the backend a reliable for GUI implementation.

1. **Project Information**

|  |  |
| --- | --- |
| Field | Value |
| Project Name | Ten-Pin Bowling Game Backend |
| System/Module | Scoring Calculation Python Class and Functions |
| UAT Start Date | 31-8-2025 |
| UAT End Date | 7-9-2025 |
| Report Date | 7-9-2025 |
| Report Version | 1.0 |

1. **Testing Scope**

**Feature to be tested**

1. Core Scoring functionality is tested, where normal frame scoring is tested without strikes and spares, Spare recognition and bonus scoring validation testing, Strike recognition and bonus scoring testing.
2. Special frame handling where the final frame with extra rolls, consecutive strikes that include multiple strikes, and correct bonus calculation.
3. Edge cases and boundary condition unit testing: where we test the perfect game unit test and gutter game test.
4. **Testing Approach**

Unit testing approach is used for individual case test.

Functional testing is also performed to verify methods and functions.

**Test environment: Python, Framework: unit test**

1. **Risk Assessment**

Edge cases may not be fully covered under the test and game.

**Test Cases Record**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step # | Step Description | Expected Result | Actual Result | Pass/Fail | Tester Notes |
| 1 | Test gutter game all zeros | Total score: 0 | Score: 0 | Pass | Correctly handles zero-pin rolls |
| 2 | Test all ones, no strikes/spares | Total score: 20 | Score: 20 | Pass | Basic scoring works correctly |
| 3 | Test single spare followed by normal roll | Score: 16 10+3+3 | Score: 16 | Pass | Spare bonus calculation is accurate |
| 4 | Test single strike followed by normal rolls | Score: 24 10+3+4+3+4 | Score: 24 | Pass | Strike bonus calculation is correct |
| 5 | Test consecutive strikes | Score: 46 24+16+6 | Score: 46 | Pass | Multiple strikes handled properly |
| 6 | Test spare in the tenth frame | Score: 13 5+5+3 | Score: 13 | Pass | Tenth frame spare rules implemented |
| 7 | Test strike in the tenth frame | Score: 17 (10+3+4) | Score: 17 | Pass | Tenth frame strike rules are working |
| 8 | Test a perfect game, all strikes | Score: 300 | Score: 300 | Pass | Maximum score calculation is correct |
| 9 | Test all spares game | Score: 150 | Score: 150 | Pass | Consistent spare scoring is accurate |

1. **Test Execution Summary after refactoring and update of the provided code**

Metric Count

Total Test Cases Planned 9

Total Test Cases Executed 9

Test Cases Passed 9

Test Cases Failed 0

Test Cases Skipped 0

Pass Rate 100%