**Bowling Game Project Summary Report**

**1. Test Summary Report Identifier**

**Report ID:** BG-TSR-v1.0  
**Project:** Bowling Game

**2. Summary**

This document provides a comprehensive overview of the testing, debugging, refactoring, and documentation performed on the ten-pin bowling game backend implementation developed for educational software.

**Items Tested:**

* BowlingGame class and all core methods
* Strike detection and bonus calculation logic
* Spare detection and bonus calculation logic
* Open frame scoring calculations
* Tenth frame special rules implementation
* Perfect game scenario handling (300 points)
* Error handling and input validation

**3. Variances**

All deviations were documented and resulted in improved code quality and test coverage.

**4. Comprehensive Assessment**

The testing process achieved significant progress against all exit criteria:

**Planned vs. Actual Coverage:**

* **Target:** 90% code coverage → **Achieved:** 95%+ code coverage
* **Target:** All business rules implemented → **Achieved:** 100% bowling rule compliance
* **Target:** No critical bugs → **Achieved:** All 5 critical bugs identified and resolved

**Quality Metrics:**

* All 15+ test cases now pass consistently
* Perfect game calculations verified (300 points)
* Complex 10th frame scenarios handled correctly
* Input validation prevents invalid game states
* Performance meets requirements for educational use

The comprehensive assessment confirms that all functional requirements have been met and the system is ready for integration with GUI components.

**5. Summary Results**

**Test Execution Results**

**Total Test Cases:** 15 comprehensive test scenarios  
**Test Categories:**

* Basic functionality: Game initialization, simple scoring (100% pass)
* Bowling rules: Strikes, spares, perfect games (100% pass)
* Edge cases: Gutter games, all spares scenarios (100% pass)
* Error handling: Invalid inputs, boundary conditions (100% pass)

**Defect Summary**

**Critical Bugs Identified and Resolved:**

1. **Incomplete Scoring Loop** (High Priority)
   * **Issue:** Original code processed only 9 frames instead of 10
   * **Fix:** Extended loop and added dedicated 10th frame handler
   * **Status:** Resolved - All games now calculate complete scores
2. **Missing Return Statement** (High Priority)
   * **Issue:** \_spare\_bonus() method was incomplete, causing runtime errors
   * **Fix:** Added proper return statement with bounds checking
   * **Status:** Resolved - All spare scenarios work correctly
3. **Incorrect Open Frame Scoring** (High Priority)
   * **Issue:** Only first roll counted in open frames, underestimating scores
   * **Fix:** Added second roll to open frame calculation
   * **Status:** Resolved - All open frames score correctly
4. **Missing 10th Frame Logic** (Critical Priority)
   * **Issue:** No handling for 10th frame special rules (bonus rolls)
   * **Fix:** Implemented complete \_score\_tenth\_frame() method
   * **Status:** Resolved - Perfect games and 10th frame scenarios work
5. **No Input Validation** (Medium Priority)
   * **Issue:** System accepted invalid pin counts (negative, >10)
   * **Fix:** Added comprehensive validation in roll() method
   * **Status:** Resolved - Invalid inputs properly rejected

**Before vs. After Comparison**

**Before Fixes:**

* Perfect Game: Failed (incomplete 10th frame)
* All Spares: Failed (missing return statement)
* Regular Games: Incorrect scores (missing rolls)
* Error Handling: No validation present

**After Fixes:**

* Perfect Game: Correct (300 points)
* All Spares: Correct (150 points)
* Regular Games: All scenarios pass
* Error Handling: Proper validation implemented

**6. Evaluation**

**Quality Assessment**

The bowling game backend now demonstrates high quality across all evaluation criteria:

**Functional Quality:**

* 100% compliance with official ten-pin bowling scoring rules
* Accurate handling of all game scenarios including edge cases
* Robust error handling prevents invalid game states
* Performance suitable for educational and recreational use

**Code Quality:**

* Comprehensive PythonDoc documentation (100% method coverage)
* Clear, readable code structure with meaningful variable names
* Proper error handling with descriptive error messages
* Efficient algorithms with minimal computational overhead

**Test Quality:**

* 95%+ code coverage across all methods and branches
* Test cases cover normal, boundary, and error conditions
* Regression test suite prevents future defects
* Clear test documentation and expected results

**7. Summary of Activities**

**Major Testing Activities**

**Planning and Design**

* Developed IEEE 829 compliant test plan with comprehensive scope
* Analyzed bowling business rules and scoring requirements
* Designed test case matrix covering all scenarios and edge cases
* Set up development environment and testing framework

**Implementation and Execution**

* Implemented 15+ unit test cases using Python unittest framework
* Executed initial test suite, identifying 5 critical bugs
* Performed systematic debugging of scoring algorithm issues
* Conducted manual verification of complex scoring scenarios

**Bug Resolution and Refactoring**

* Fixed all identified critical bugs with proper error handling
* Refactored code for improved readability and maintainability
* Added comprehensive PythonDoc documentation to all methods
* Enhanced test suite with additional edge cases and regression tests

**Documentation and Validation**

* Generated complete API documentation using PythonDoc
* Performed final validation testing of all scenarios
* Created professional project documentation and reports
* Established Git repository with meaningful commit history

**8. Approvals**

|  |  |
| --- | --- |
| Role | Name |
| Test Lead | Piyush Tatwani |
| Course Tutor | Sana Alyaseri |

**Test Results Summary**

* All 15 test cases: PASSED
* Code coverage: 95%+
* Performance: Meets requirements
* Documentation: Complete