**Bowling Game Testing & Maintenance – Summary Report**

**Name:** [Your Full Name]  
**Course:** IT6039 – Software Testing & Maintenance **Project Title:** Bowling Game Backend – Test & Refactor **Submission Date:** 21-Apr-25  
**GitHub Repository Link:** https://github.com/arsalansm2/BowlingGame

**Testing Approach**

A white-box testing strategy was adopted to verify the correctness of a 10-pin bowling game backend implemented in Python. A comprehensive suite of unit tests was designed using the unit test module. Test cases covered:

* Basic rolls
* Spares and strike logic
* Edge cases (perfect game, all gutter balls)
* Tenth frame bonus rules

Tests were grouped using setUp() and helper methods like roll\_many(), enabling reusable and readable test design.

**Debugging & Fixes**

During testing, a potential bug was identified where the scoring function could fail if insufficient rolls were recorded, resulting in IndexError. This was resolved by introducing a helper method safe\_get() to safely access elements beyond the current index, returning 0 if out of bounds. This ensured robust bonus calculation for strikes and spares.

**Code Refactoring**

The scoring loop was refactored to:

* Use safe\_get() for safe access
* Improve variable naming and logic clarity
* Simplify frame logic using clean, modular helper functions

Refactoring improved code readability, reduced the risk of bugs, and followed Python best practices.

**Version Control with Git**

Version control was managed using Git. Key milestones (initial logic, test cases, bug fix, refactor, documentation, and report writing) were committed with clear messages. The .git folder and GitHub repository contain full commit history and documentation of the workflow.

**Documentation**

Each method was documented using PythonDoc-style comments. Documentation was generated with pydoc, producing a browsable HTML API file. This improves maintainability and supports future development phases.

**Conclusion**

All learning outcomes were achieved:

* LO1, LO2: Through thorough test design and implementation
* LO4: Debugging and performance-aware refactoring
* LO5: Use of Git and GitHub for version control
* LO6: Full inline and external documentation

The project backend is now stable, tested, documented, and ready for integration with GUI or database input layers.

**Code Refactoring**

def score(self):

"""Calculate total score for the game."""

result = 0

roll\_index = 0

for frame in range(10):

if self.is\_strike(roll\_index): # Strike

result += 10 + self.strike\_bonus(roll\_index)

roll\_index += 1

elif self.is\_spare(roll\_index): # Spare

result += 10 + self.spare\_bonus(roll\_index)

roll\_index += 2

else: # Open frame

result += self.sum\_of\_balls\_in\_frame(roll\_index)

roll\_index += 2

return result

Above code is refactored as below to improve the readability.

def score(self):

result = 0

roll\_index = 0

for \_ in range(10): # 10 frames

if self.is\_strike(roll\_index):

result += 10 + self.strike\_bonus(roll\_index)

roll\_index += 1

elif self.is\_spare(roll\_index):

result += 10 + self.spare\_bonus(roll\_index)

roll\_index += 2

else:

result += self.sum\_of\_balls\_in\_frame(roll\_index)

roll\_index += 2

return result