# Bowling Game Project – Summary Report

## Overview

This project was about testing, fixing, and improving a Python program that calculates scores for a ten‑pin bowling game. The main goal was to check if the program followed the bowling rules, find any bugs, write tests, and make the code easier to read and use.

## Testing Approach

I tested 10 different bowling game situations. The table below shows each test case, what it checks, and the expected result. I also left a space where I will add screenshots of the test run.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expected Result | Screenshot |
| 1. All Gutters | Roll 20 times with 0 pins | Score = 0 | [Screenshot here] |
| 2. All Ones | Roll 20 times with 1 pin | Score = 20 | [Screenshot here] |
| 3. One Spare | Spare followed by 3 pins | Score = 16 | [Screenshot here] |
| 4. One Strike | Strike followed by 3 and 4 pins | Score = 24 | [Screenshot here] |
| 5. Perfect Game | 12 strikes | Score = 300 | [Screenshot here] |
| 6. 10th Frame Spare | Spare in 10th with bonus 7 | Score = 17 | [Screenshot here] |
| 7. 10th Frame Strike | Strike in 10th with 7 and 2 | Score = 19 | [Screenshot here] |
| 8. No Roll After Finish | Try to roll after game ended | Raise ValueError | [Screenshot here] |
| 9. Invalid Negative Roll | Roll -1 pins | Raise ValueError | [Screenshot here] |
| 10. Invalid Too Many Pins | Roll 11 pins | Raise ValueError | [Screenshot here] |

## Debugging & Fixes

When running the tests, I noticed the code could sometimes crash with an index error when checking for strike or spare bonuses. I fixed this by making the strike and spare bonus calculations safer so they don’t go outside the list of rolls.

## Refactoring

I made the code easier to read and safer by:  
- Adding type hints (like int and bool)  
- Writing clear comments and docstrings  
- Moving strike and spare logic into helper methods  
- Using underscores for helper methods to show they are internal  
- Organising the code into clear sections

## Version Control (Git)

I used Git to track changes. I made commits after writing tests, fixing problems, refactoring the code, and adding documentation. Commit messages explained clearly what was changed.

## Documentation

I added docstrings to all methods and generated documentation using pydoc with the command:  
python -m pydoc -w bowling\_game  
This created a file bowling\_game.html that explains the code in a simple way.

## Conclusion

The bowling game program is now fully tested and works correctly. It follows the bowling rules, handles edge cases, and has clear documentation. The code is easier to understand and ready for future improvements like adding a user interface.