# Test Plan for bowling game

## App Description

A console-based program called The Python Bowling Game simulates the game of ten-pin bowling. Users can roll a bowling ball and the program will calculate their score based on how many pins they knock down in each frame. The game adheres to the traditional bowling rules, which include strikes, spares, and bonus throws in the last frame. The Python-based application's goal is to deliver great bowling experience.

## Features to be tested with test cases

### • Score Calculations

Check to see if the game accurately computes the score based on the bowling rules in a variety of situations:

Gutter game – (testGutterGame)

1. Roll 0 pins for all 20 rolls.
2. Verify that the final score is 0.
3. Reasoning: This test ensures that the game correctly calculates a score of 0 for a game with no pins knocked down.

All ones – (testAllOnes)

1. Roll 1 pin for all 20 rolls.
2. Verify that the final score is 20.
3. Reasoning: This test checks if the game correctly calculates the score for a game where 1 pin is knocked down in each roll.

One spare – (testOneSpare)

1. Roll 5 pins, 5 pins, and 3 pins for the first three rolls.
2. Roll 0 pins for the remaining 17 rolls.
3. Verify that the final score is 16.
4. Reasoning: This test verifies the game's ability to handle spares and calculate the correct score.

One strike – (testOneStrike)

1. Roll 10 pins, 4 pins, and 3 pins for the first three rolls.
2. Roll 0 pins for the remaining 16 rolls.
3. Verify that the final score is 24.
4. Reasoning: This test ensures that the game correctly calculates the score for a game with a strike and additional rolls.

Perfect game – (testPerfectGame)

1. Roll 10 pins for all 12 rolls.
2. Verify that the final score is 300.
3. Reasoning: This test checks if the game handles a perfect game with all strikes and awards the maximum score.

Many spares – (testManySpare)

1. Roll 5 pins for all 21 rolls.
2. Verify that the final score is 150.
3. Reasoning: This test examines the game's ability to calculate scores when spares are rolled consistently.

### • Input validation

Confirm that the game handles invalid inputs gracefully

* Verify that rolling more than ten pins in a single frame is not allowed.
* Check if rolling more than the maximum number of frames is prevented.

### • Game state management

* Confirm that the frame number is incremented after each roll.
* Verify that the total score is updated correctly after each roll.

## Features not to be tested

### • UI

UI is out of scope as the application doesn’t have one at the moment and it will be covered in the separate testing phase.

### • Multiple players

This feature is not developed yet, thus out of scope for the testing. Will be covered in the separate testing phase.

## Execution

### • Environment config

The required Python interpreter and dependencies have been installed in a controlled and isolated testing environment. The unittest framework has been integrated with the Bowling Game code to enable automated testing.

### • Assessment

The Bowling Game has undergone extensive testing that included input validation, game state management, and scoring calculation. With no defects found, all provided unittests and additional test cases have been executed successfully.

## • Recommendations

It is advised to move forward with confidence in releasing the Bowling Game to users based on the successful completion of all test cases and the absence of bugs. The Bowling Game provides examples of accurate game state management, proper input validation, and correct scoring calculation. Reviewing the test reports will help stakeholders determine whether the product is ready for release.