Test Plan – Bowling Game

Name: - Kartik Bansal

Date: 2025-09-04

# 1) Purpose

This plan explains, in simple terms, how we will test a small Python program that scores a 10‑pin bowling game. The goal is to prove the scoring rules are correct and easy to maintain.

# 2) What We Are Testing (Unit‑Test Game)

Program: one class called BowlingGame in bowling\_game.py.  
Functions under test:  
• roll(pins): record a roll (0–10). Must reject invalid values.  
• score (): total score for a full game of 10 frames.

Bowling rules we must follow:  
• Open frame: add the two rolls in that frame.  
• Spare (10 pins in two rolls): 10 + next one roll.  
• Strike (10 pins in first roll): 10 + next two rolls.  
• 10th frame: allow bonus roll(s) if strike/spare happens.

# 3) Scope

In scope: scoring rules, frame progression, bonus handling, and input validation.

Out of scope: user interface, file/database input, multiple players, performance tuning.

# 4) Test Strategy

We use simple, automated unit tests with pytest. Each test feeds a known sequence of rolls into BowlingGame and checks the final score. We also test invalid input to make sure roll() raises errors.

# 5) Test Data

We use small, memorable scenarios so anyone can follow the math:  
• Gutter game (all 0s) → score 0  
• All ones (20×1) → score 20  
• One spare then a 3 → 10 + 3 + 3 = 16  
• One strike then 3 and 4 → 10 + (3+4) + 3 + 4 = 24  
• All spares (21×5) → 150  
• Perfect game (12×10) → 300  
• Regular no marks [3,4,2,5,1,6,...] → 72  
• Mixed example from brief → 190  
• Invalid inputs: −1, 11, 3.5 must raise ValueError

# 6) How To Run The Tests

1. Open the folder in VS Code and open a terminal.  
2. Create a virtual environment and install tools:  
 python -m venv .venv  
 .venv\Scripts\activate  
 pip install pytest pytest-cov  
3. Run the sample scenarios:  
 python run\_scenarios.py  
4. Run the tests and coverage:  
 python -m pytest -q  
 python -m pytest --cov=bowling\_game --cov-report=term-missing

# 7) Passing Criteria

• All tests pass (green in pytest).  
• Line coverage is at least 95% (aim for 100%).  
• No unexpected errors or crashes.

# 8) Risks & Mitigation

• Mistakes in 10th frame bonuses → we include tests for perfect and all‑spares.  
• Wrong strike/spare math → we include unit tests that check the next one/two rolls.  
• Import path errors during testing → tests/conftest.py adds the project root to sys.path.