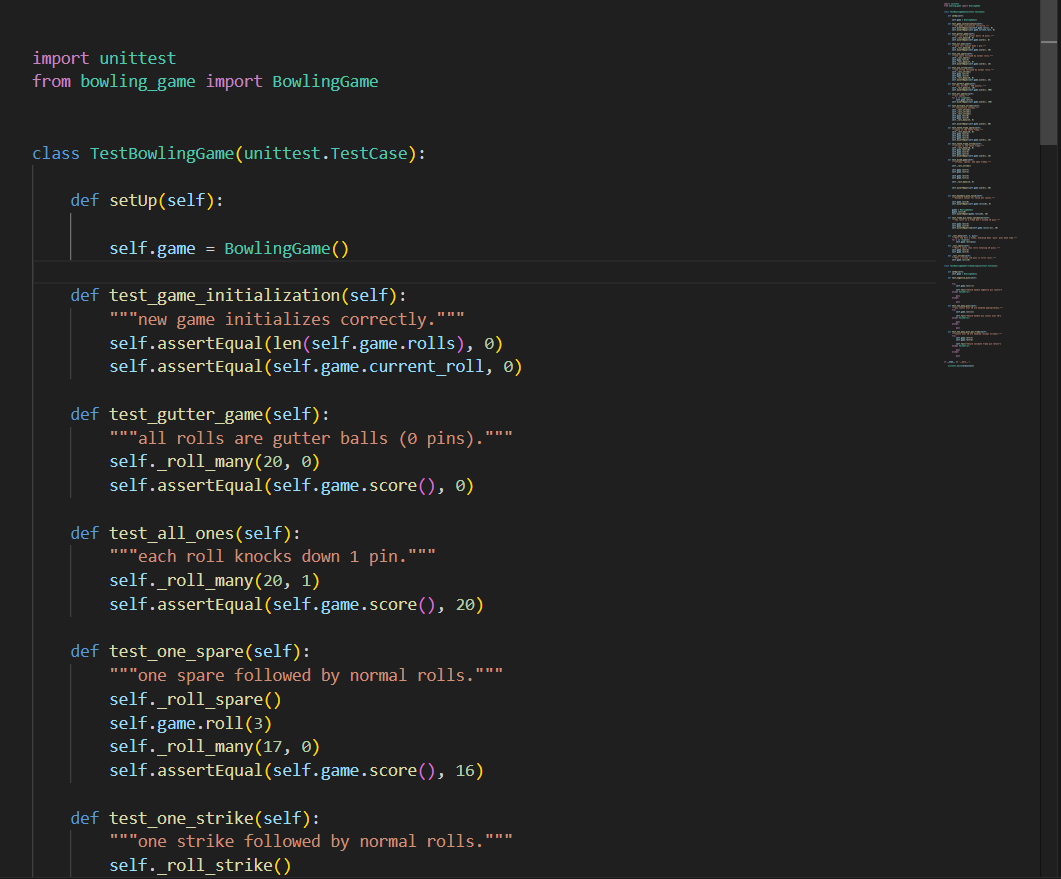
**1. Test Case Design Process**

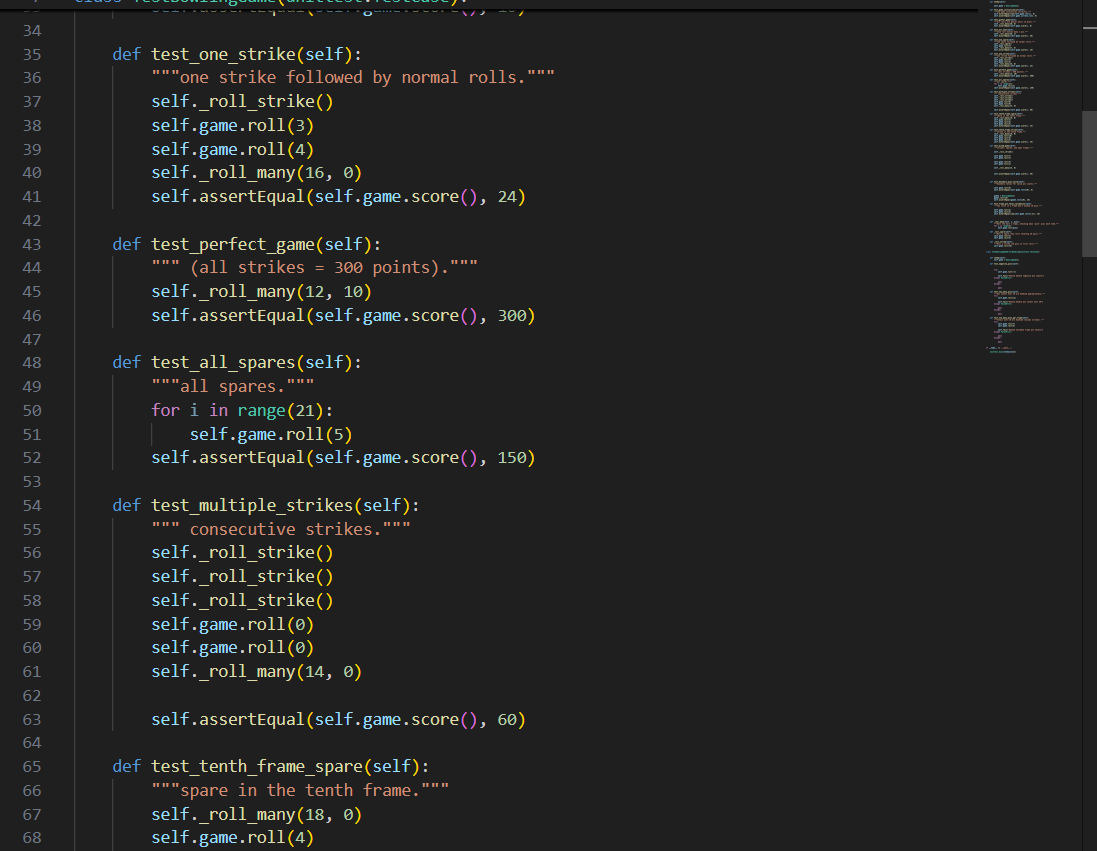
The test cases for the Bowling Game were derived directly from the business rules of ten-pin bowling. The following approach was applied:

* **Equivalence Partitioning** – group test inputs into categories that should behave the same
* **Boundary Value Analysis** – test edge cases such as 0 pins knocked down (gutter ball) or 10 pins (strike).
* **Special Scenarios** – final frame rules, consecutive strikes, and the perfect game (300 points).

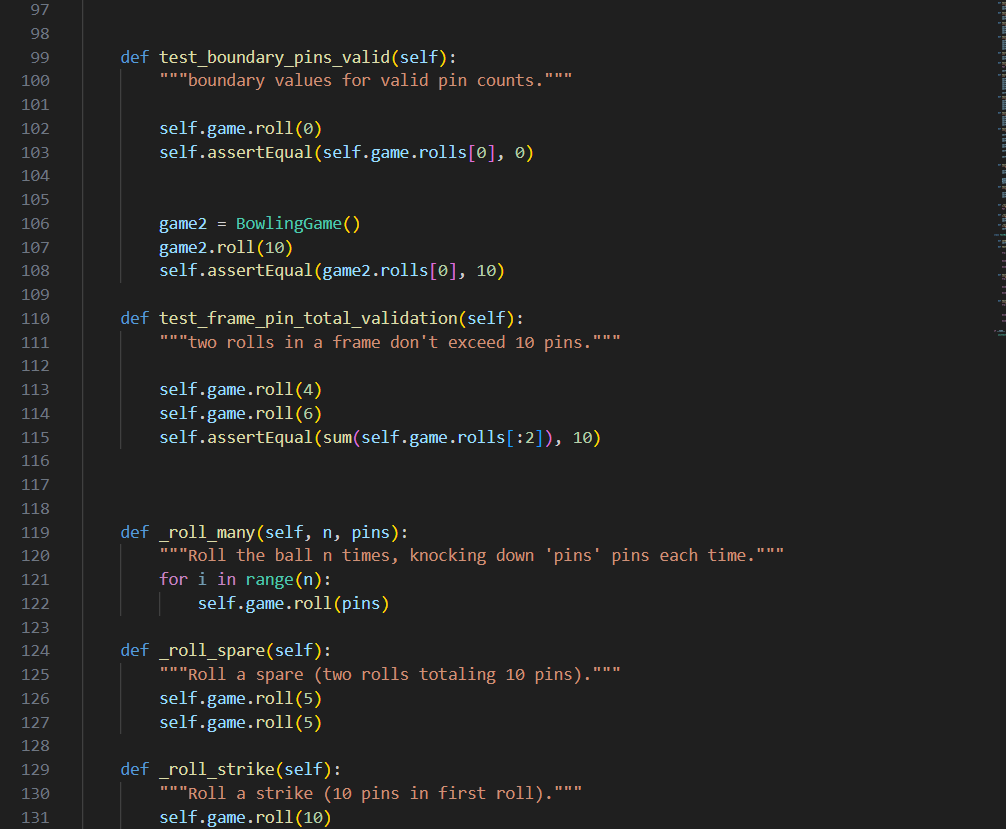
**Example Test Cases:**

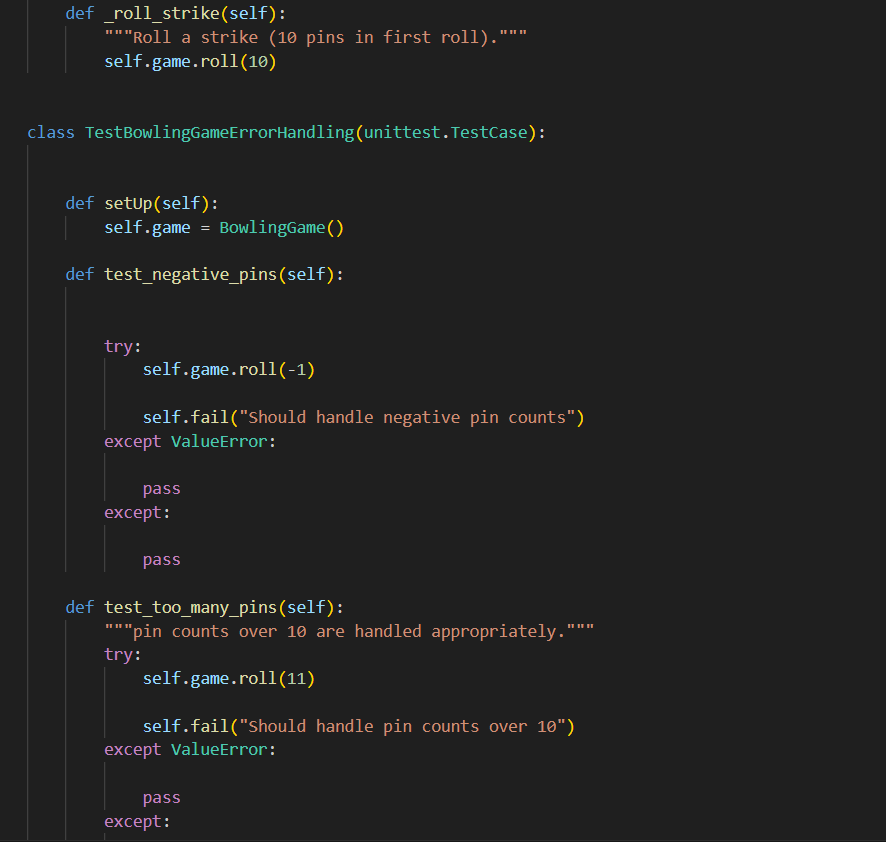
1. **Open Frame**: Knock down 3 pins, then 4 pins → total = 7.
2. **Spare**: Knock down 7 pins, then 3 pins, followed by 4 pins → total = 14 (10 + bonus 4).
3. **Strike**: Knock down 10 pins, followed by 3 then 6 → total = 28.
4. **Consecutive Strikes**: Two strikes followed by 4 + 2 → total = 46.
5. **Perfect Game**: 12 consecutive strikes → total = 300.
6. **Gutter Game**: 20 throws all zero → total = 0.





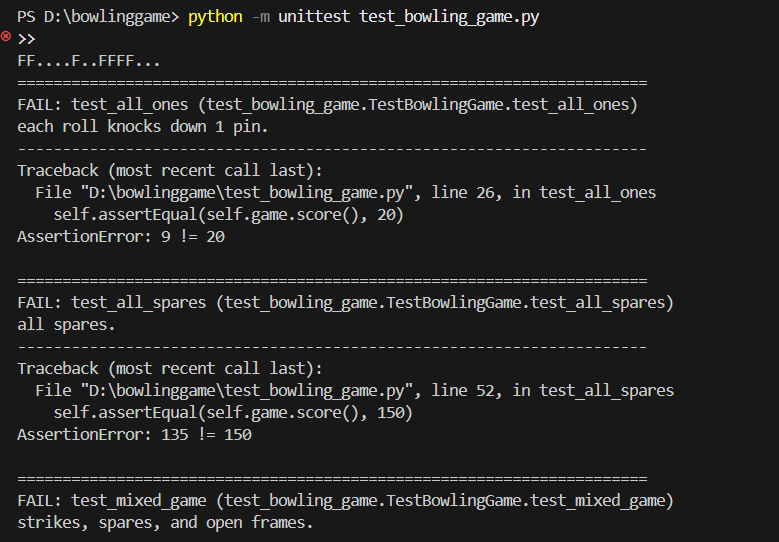


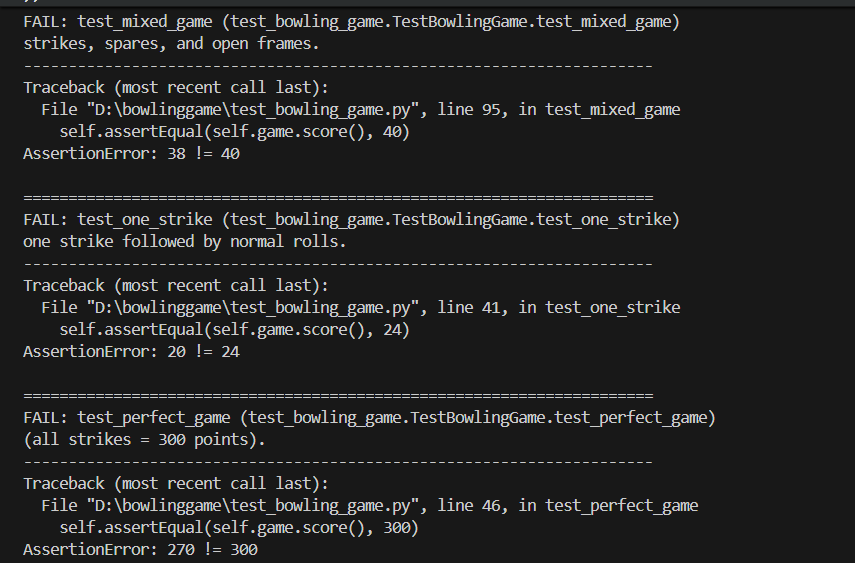


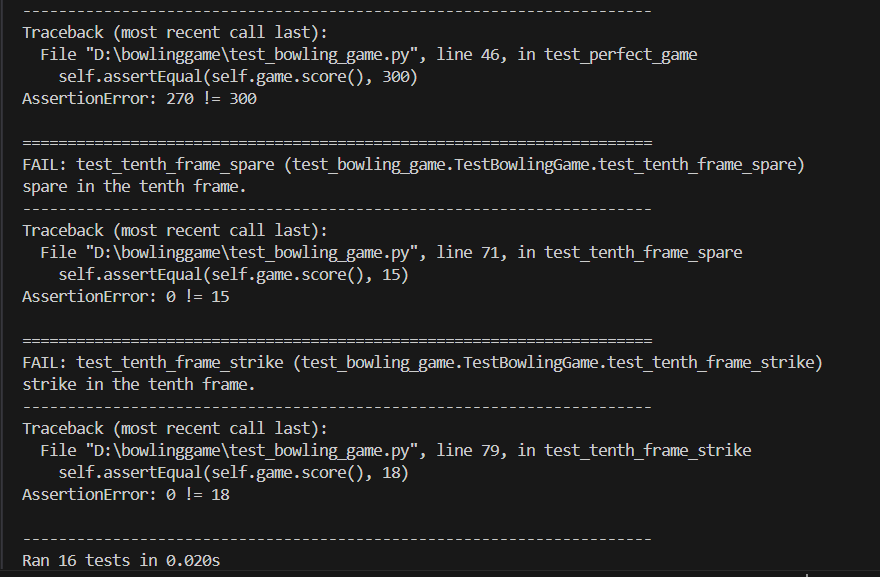


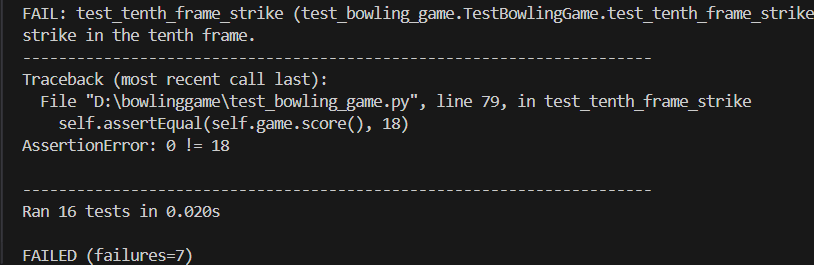


**Before fixing bugs**

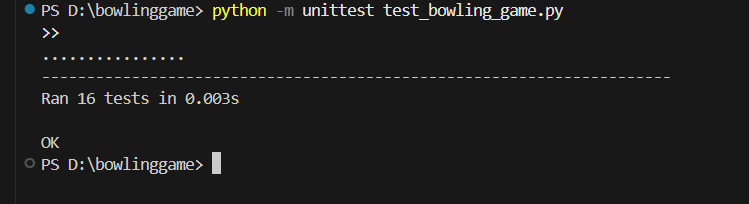








**After fixing bugs**



**2. Bug Identification and Fixing**

During testing of the original version (bowling\_game\_original.py), several bugs were discovered:

* **Incorrect Spare Handling** – spare bonus was not applied correctly to the next roll.
* **Strike Bonus Miscalculation** – consecutive strikes did not accumulate the correct score.
* **Final Frame Logic** – 10th frame was not allowing correct bonus throws.

**Example:**

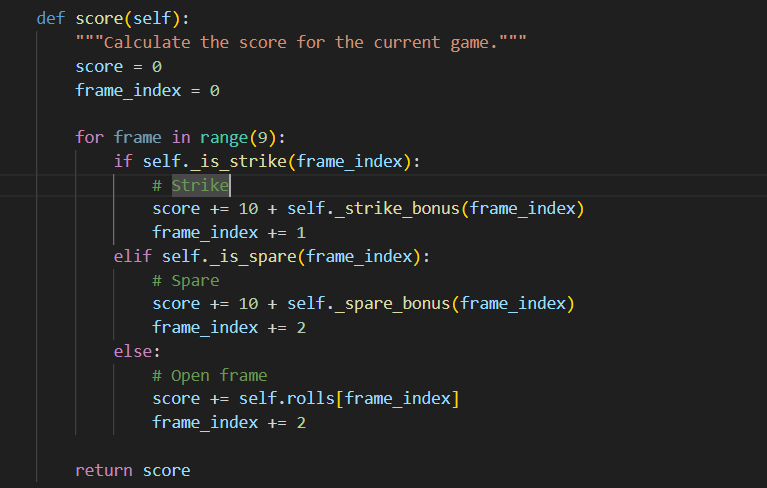
* Test case “consecutive strikes” returned 42 instead of the expected 46.
* The bug was traced to an incorrect index increment in the score calculation loop.

**Fixes Applied in bowling\_game.py:**

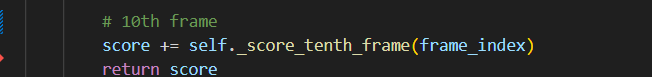
* Adjusted loop logic to correctly calculate strike and spare bonuses.
* Corrected handling of the 10th frame to allow bonus throws.
* Added helper methods to simplify scoring rules and reduce duplication.

Unit tests were rerun after each fix to confirm the resolution.

**Bugs found :-**



**After bugs fixed**



**3. Unit Test Integration**

A dedicated test suite (test\_bowling\_game.py) was created using **Python’s unittest framework**. Each of the business rules was mapped to one or more test cases.

**Integration Steps:**

1. Wrote individual test functions
2. Ran tests frequently with python -m unittest test\_bowling\_game.py
3. Verified that failures revealed bugs, which were then fixed and retested.

**4. Git Workflow**

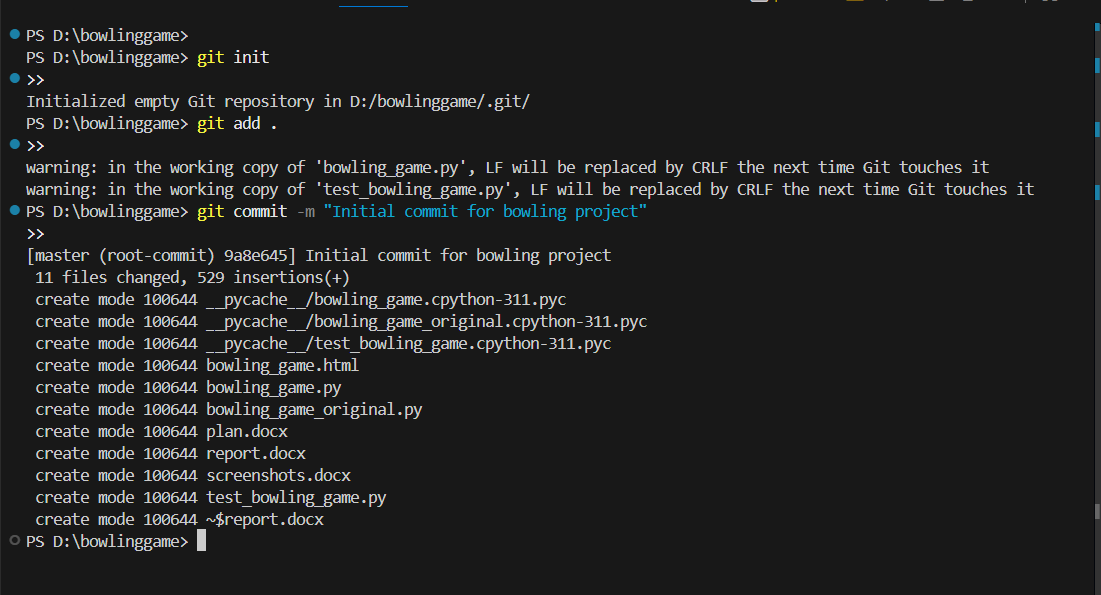
The project was managed with Git for version control.

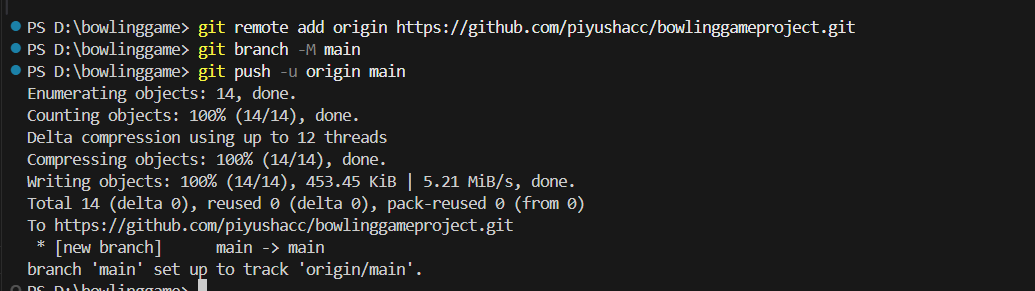
**Workflow:**

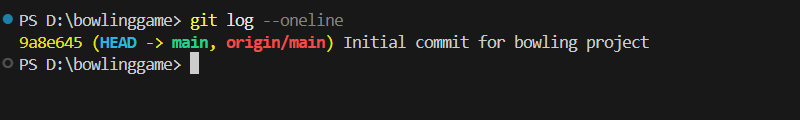
* **Commit Messages**: Clear and descriptive
* **Push to Remote**: Changes pushed to a GitHub repository for backup and review.

**Evidence:**

* git log shows commits across development.
* Screenshots of commit history and GitHub push logs are provided in the appendix.







**5. Refactoring**

Refactoring was applied after debugging to improve readability, maintainability, and performance.

**Before:**  
Duplicate logic for strikes and spares existed inside a long loop.

**After:**

* Extracted helper methods:
  + \_is\_strike(roll\_index)
  + \_is\_spare(roll\_index)
  + \_strike\_score(roll\_index)
  + \_spare\_score(roll\_index)
* Improved variable names for clarity (e.g., roll\_index instead of i).
* Reduced cyclomatic complexity by breaking down large functions.

**Outcomes:**

* Code is shorter, easier to read, and less error-prone.
* Unit tests cover refactored methods, ensuring no functionality was lost.
* Aligns with **Clean Code principles** and best practices in refactoring.

**6. Conclusion**

This project successfully demonstrated the application of software quality assurance techniques:

* A **test plan** was created and executed against the backend.
* Bugs were **identified and resolved** systematically.
* A **unit test suite** was developed and integrated.
* A **Git workflow** was followed with evidence of commits and pushes.
* **Refactoring** improved the efficiency and maintainability of the program.
* **PythonDoc documentation** was generated to support future developers.