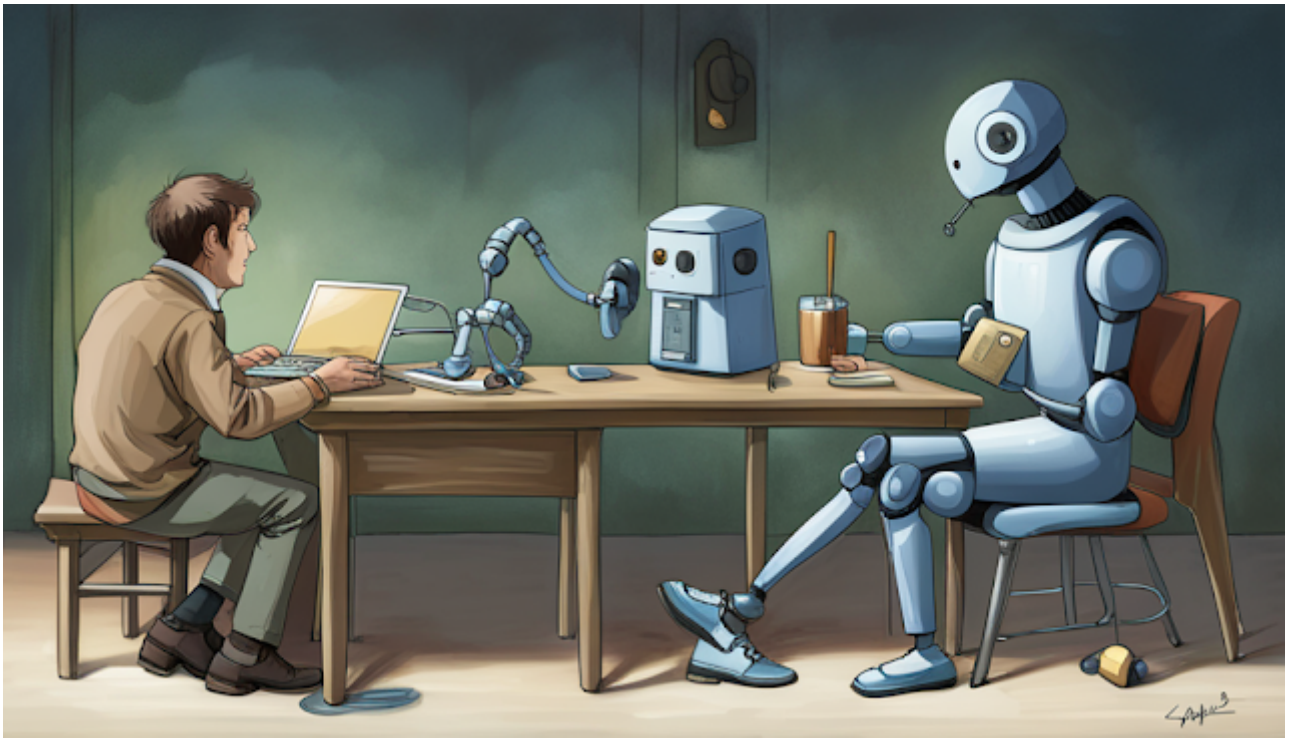
AI Muggins

04 July 2023 · ai, automation, bias, learning

I play a card game called cribbage. I often play it with my son (https://www.investigatingsoftware.co.uk/2020/04/dealing-with-bugs-using-impossible-tests.html). One interesting part of the game is the muggins rule. This means that you can claim points from other players turns, if they miscount the score.

The scoring is slightly nerve racking, with each of us double and triple checking our scores, to avoid falling foul of ‘muggins’, that’s part of the fun.



But my son and I also find ourselves discussing other hands of cards, in a sort of alternate history version of the game. “So if I had a 7 instead of a 2 of hearts, then I’d get a double run and score at least 8 more points”.

“Yes Dad, if you had different cards then you would likely have a different score, but you don’t” he says while rolling his eyes.

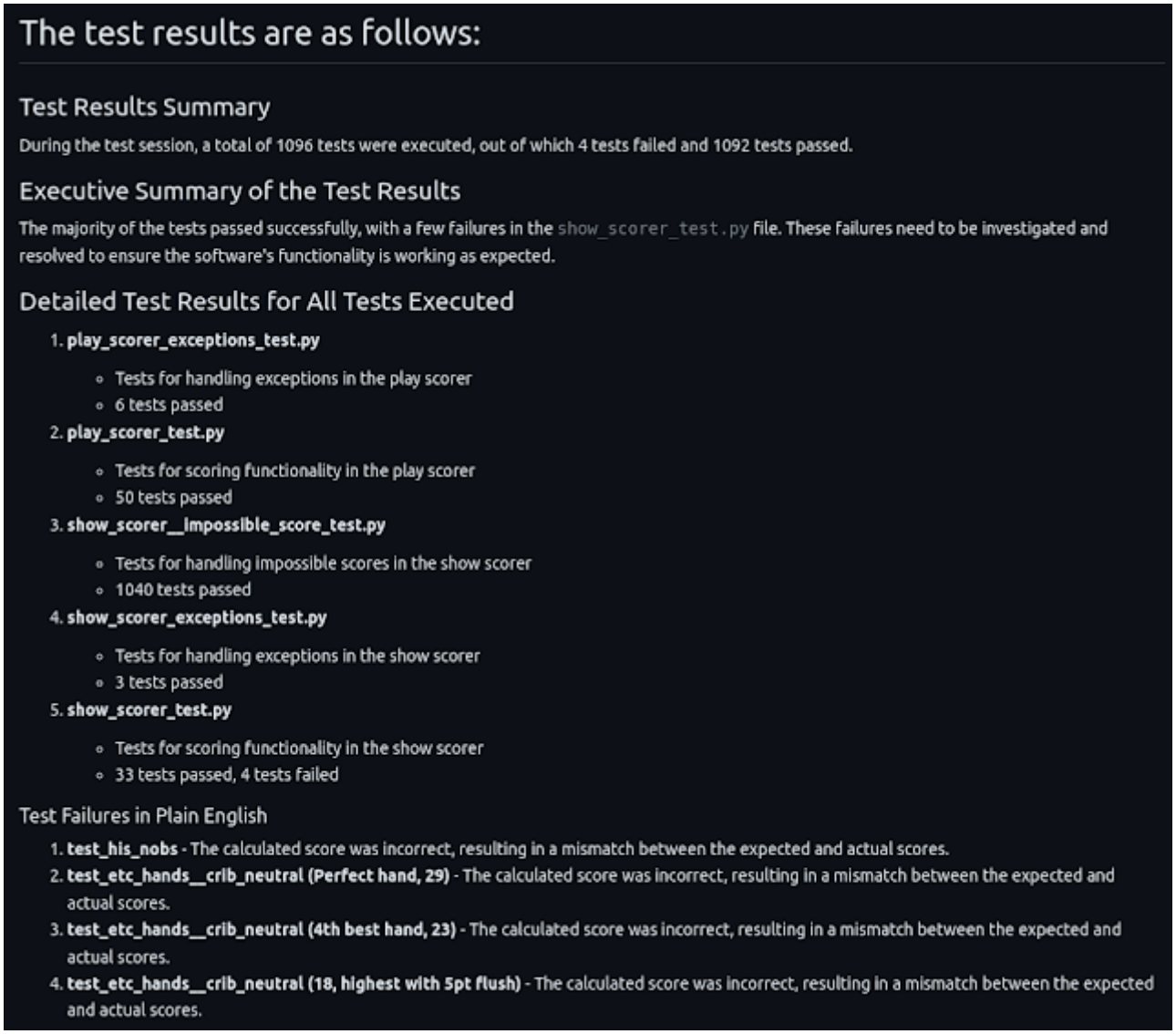
This sort of bitter-sweet history rewriting is a convenient tool for us to swallow the awkward truth of the real world. We often create alternate things to object to.

Take Chat GPT 4 and tools like Copilot X. These are powerful tools, capable of doing useful tasks quicker and more easily than other tools.

But no... (People say), they are dangerously sentient, or not sentient, fake, poor at this one task or too good at some other thing that people get paid for. To paraphrase my son, “Yes, if they were X then they would not be Y”.

These tools are, well..., Tools. They have their limits that we are still discovering – they also have great abilities which we are also only just realising. Unlike our existing tools we haven’t had a chance to evaluate them and find their place. A task made harder by the fact that AI technology is improving extremely fast at the moment.

Just in the realm of software test automation alone there are many opportunities where GPT4 and Copilot could help. For example, summarising test results, and providing the results in a human readable form. E.g.:



The test results are as follows:

Test Results Summary

During the test session, a total of 1096 tests were executed, out of which 4 tests failed and 1092 tests passed.

Executive Summary of the Test Results

The majority of the tests passed successfully, with a few failures in the `show_scorer_test.py` file. These failures need to be investigated and resolved to ensure the software's functionality is working as expected.

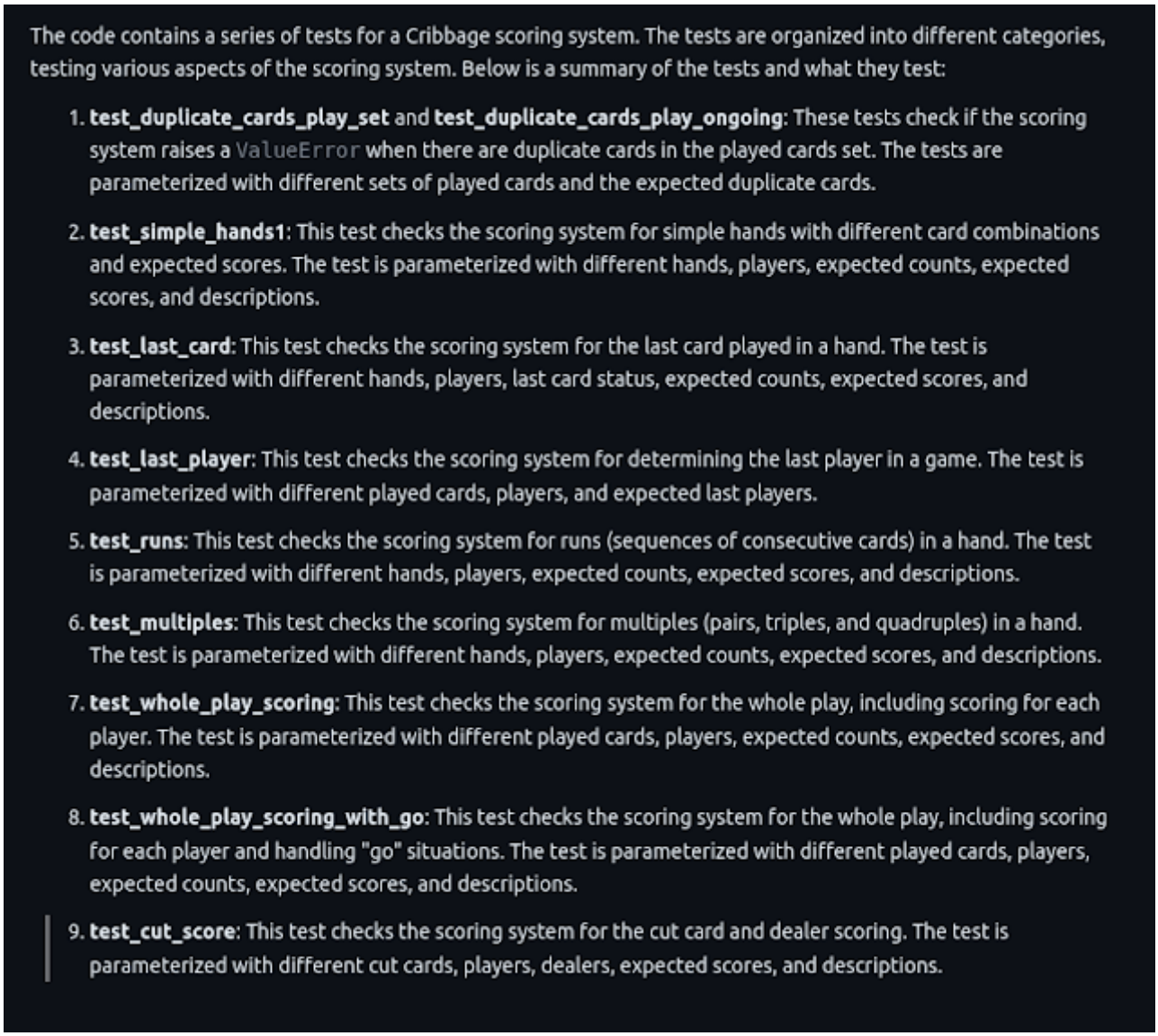
Detailed Test Results for All Tests Executed

1. **play_scorer_exceptions_test.py**
   - Tests for handling exceptions in the play scorer
   - 6 tests passed
2. **play_scorer_test.py**
   - Tests for scoring functionality in the play scorer
   - 50 tests passed
3. **show_scorer__impossible_score_test.py**
   - Tests for handling impossible scores in the show scorer
   - 1040 tests passed
4. **show_scorer_exceptions_test.py**
   - Tests for handling exceptions in the show scorer
   - 3 tests passed
5. **show_scorer_test.py**
   - Tests for scoring functionality in the show scorer
   - 33 tests passed, 4 tests failed

Test Failures in Plain English

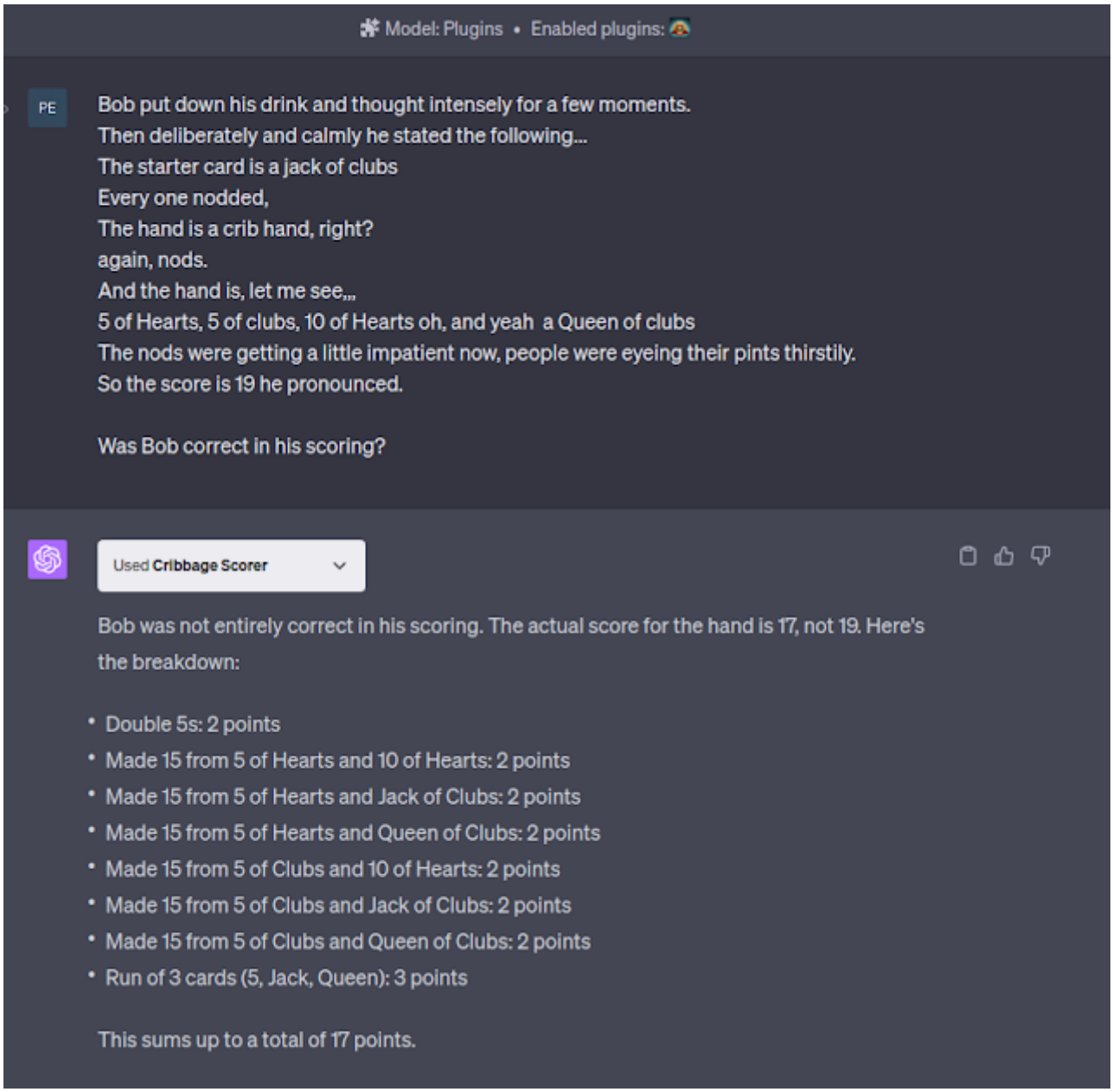
1. **test_his_nobs** - The calculated score was incorrect, resulting in a mismatch between the expected and actual scores.
2. **test_etc_hands__crib_neutral (Perfect hand, 29)** - The calculated score was incorrect, resulting in a mismatch between the expected and actual scores.
3. **test_etc_hands__crib_neutral (4th best hand, 23)** - The calculated score was incorrect, resulting in a mismatch between the expected and actual scores.
4. **test_etc_hands__crib_neutral (18, highest with 5pt flush)** - The calculated score was incorrect, resulting in a mismatch between the expected and actual scores.

Or explaining test code, without the need for cumbersome abstraction layers like cucumber, E.g.:



This is a GPT4 API interpretation of the tests for my cribbage scorer.

The creation of basic unit tests for existing code to enable easy refactoring, or when combined with ‘function calling’ being able check the results contained in a body of text. E.g,:



This is my Cribbage Scoring Plugin, available in ChatGPT4.

Testers & Test engineers often fall into the “Ha it can’t do this” school of thought with new tools. Rather than thinking I’ve been given access to a particularly useful text and code analysis and generation tool for a price that approaches free.

A tool that is improving month by month, a tool that extends my reach and increases my performance (compared to those shunning it because it couldn’t do some party trick or a skill you’ve spent your career honing.